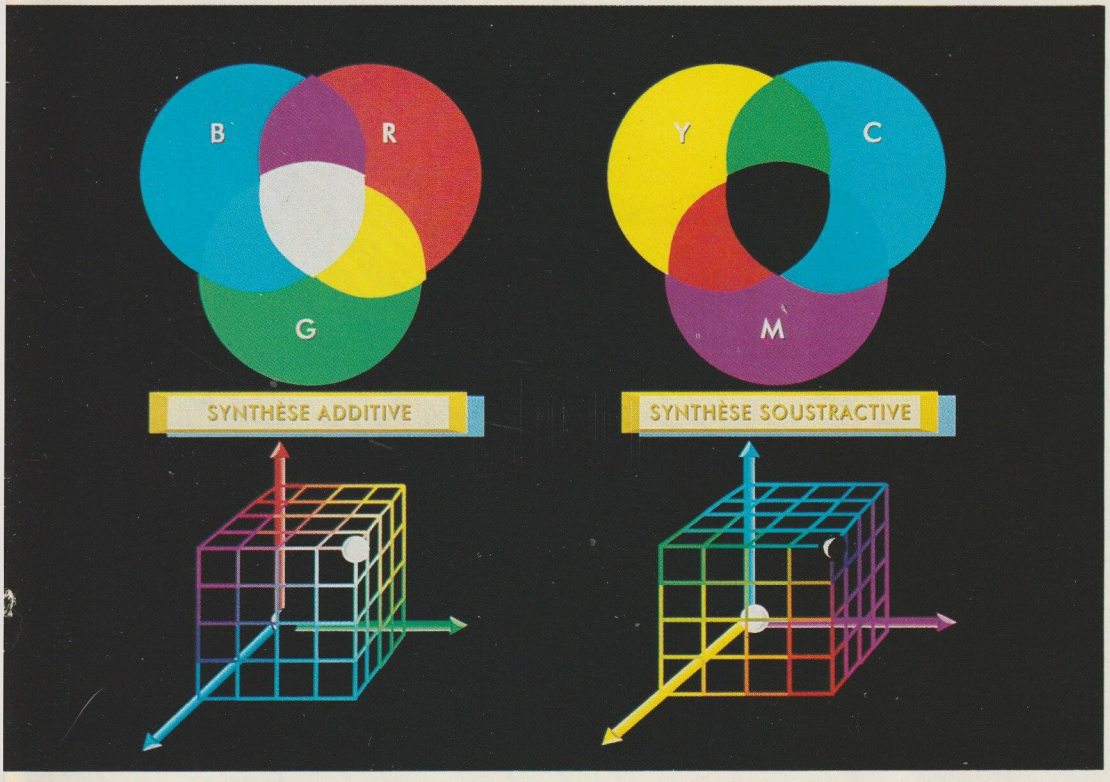
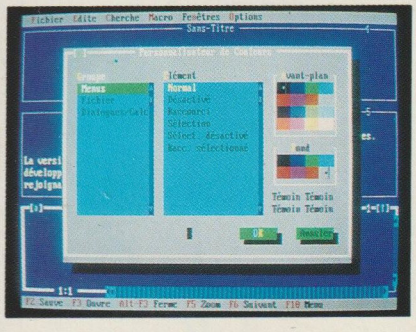
INFOPC
langages & systèmes

MARS 1991 - Nº 1
SUPPLÉMENT
DÉVELOPPEURS
DU N º 69 D'INFOPC



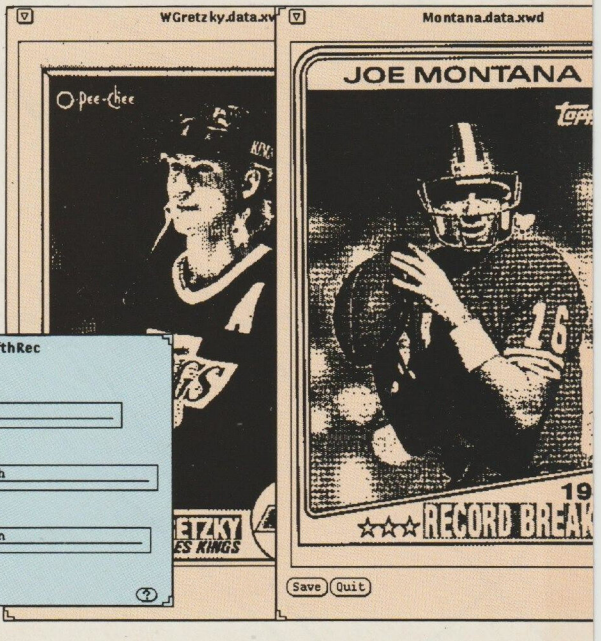
LA GESTION DE LA COULEUR

■ Principes et mise en œuvre de l'image numérique



TURBO VISION

■ La bibliothèque objet de Turbo Pascal 6.0

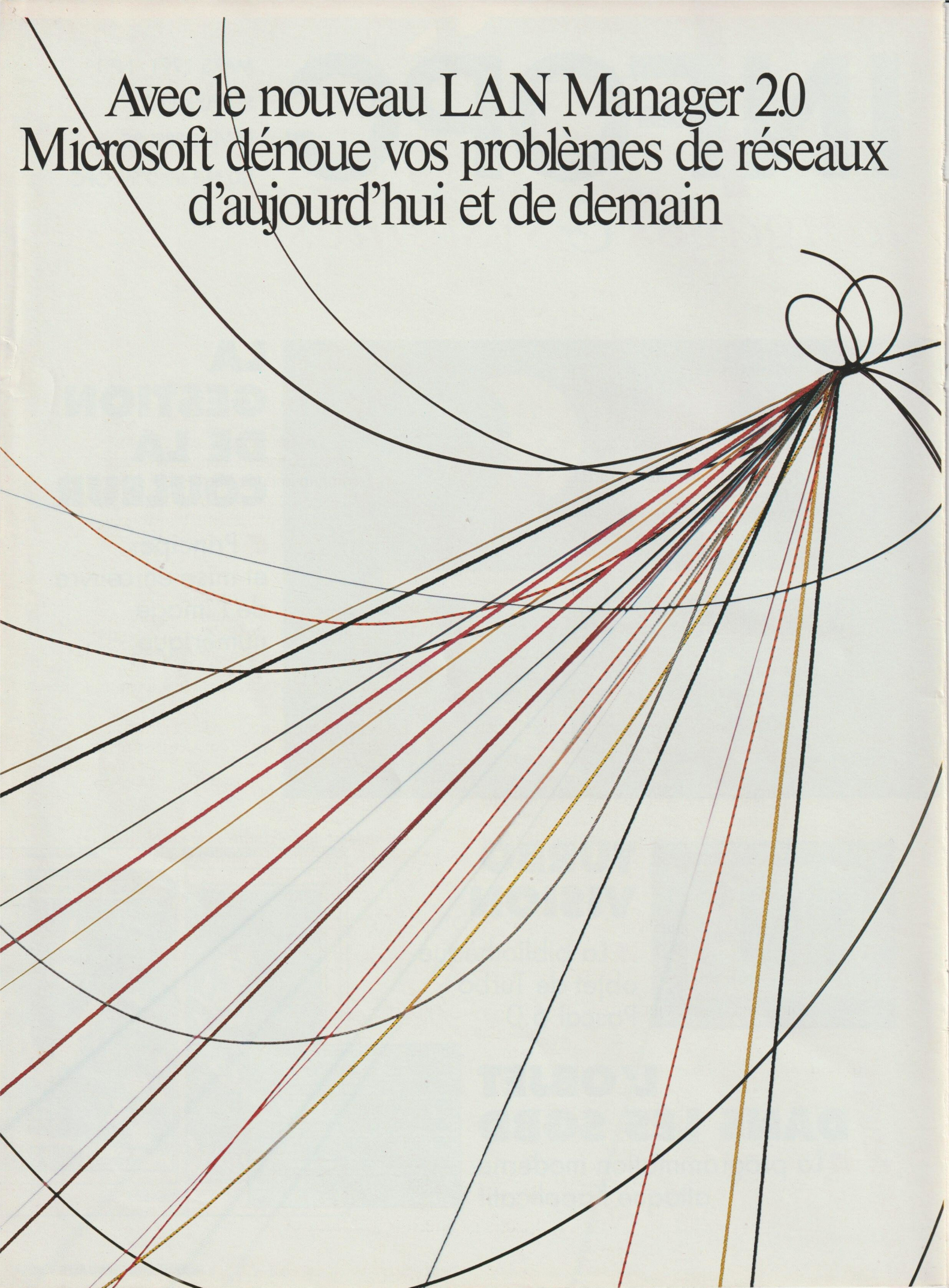


L'OBJET DANS LES SGBD

■ La programmation moderne attaque l'applicatif

NE PEUT ÊTRE VENDU SÉPARÉMENT

Avec le nouveau LAN Manager 2.0 Microsoft dénoue vos problèmes de réseaux d'aujourd'hui et de demain



Microsoft LAN Manager 2.0, le système d'exploitation réseau conçu par le leader mondial du logiciel.

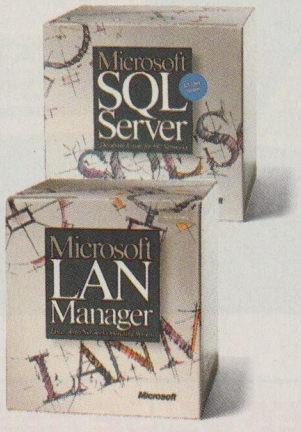
La décennie 90 sera caractérisée par le thème de la connexion. En annonçant aujourd'hui LAN Manager 2.0, Microsoft vous offre une évolution progressive de votre réseau, du simple partage de ressources locales jusqu'à l'intégration complète du poste de travail au réseau d'entreprise.

L'impulsion Microsoft vers l'architecture Client-Serveur.

Après avoir contribué à l'avènement d'une informatique plus conviviale, Microsoft met le cap sur l'architecture Client-Serveur avec LAN Manager. Tout en restant compatible avec les applications réseaux basées sur le standard Netbios, Microsoft LAN Manager 2.0 constitue l'environnement de développement d'une nouvelle génération d'applications basées sur la répartition des traitements entre postes clients sous MS-DOS, Microsoft Windows ou OS/2 et postes serveurs intelligents, division des tâches appelée également architecture Client-Serveur.

Une plate-forme système sûre et performante.

De la simple reprise de l'existant en matière de partage de ressources jusqu'à la répartition des traitements entre plusieurs postes multi-processeurs, LAN Manager 2.0 vous offre une plate-forme système dotée d'outils d'administration, de sécurité et d'un niveau de performance apte à répondre aux exigences de vos applications de production. Basé sur un environnement d'exploitation sûr et protégé : OS/2 et un système de gestion de fichiers 32 bits à haute performance : HPFS 386, LAN Manager exploite pleinement les capacités des micro-ordinateurs les plus performants du marché. L'administration de plusieurs serveurs est facilitée avec la notion de domaine. La sécurité des données est assurée grâce à des outils complets de back-up, de disques en miroir ou en duplex.



La naissance d'une nouvelle gamme de produits : la série réseau.

Microsoft SQL Server 1.1, DCA/Microsoft Communications Server ou ATLANTIS/Microsoft Communications Server sont les premiers constituants de cette série réseau : moteur de bases de données et serveur de communications. Microsoft SQL Server 1.1 représente aujourd'hui l'état de l'art en matière de traitement transactionnel. Grâce au support de nombreuses applications frontales telles que Microsoft Excel, Lotus 1-2-3 ou Borland Paradox..., une nouvelle combinaison gagnante est désormais offerte à l'informatique de groupe d'aujourd'hui et de demain en matière d'accès et de manipulation de données.

Un standard adopté par les principaux constructeurs informatiques.

LAN Manager a été élu par les principaux constructeurs tels qu'IBM, 3 Com, Hewlett-Packard, AT&T, DEC, Ungermann-Bass, Olivetti, Nokia, SMT-Goupil et bien d'autres pour constituer la base de leur offre réseau local.

Un support de proximité à valeur ajoutée : le "Revendeur Spécialiste Réseau Microsoft."

Afin d'assurer le meilleur niveau de support et de prestations aux entreprises, les "Revendeurs Spécialiste Réseaux Microsoft" offrent les meilleures garanties de services en matière de connectivité et de bases de données pour la mise en place de Microsoft LAN Manager 2.0 et/ou Microsoft SQL Server 1.1.

Pour recevoir la liste des revendeurs spécialistes réseaux Microsoft ou des informations commerciales sur les produits de la série réseaux, envoyez-nous votre carte de visite à Microsoft France, Marketing Réseaux, 91957 Les Ulis CEDEX ou appelez-nous au : Service Clients 69 86 46 46.

Microsoft®

Nous civilisons la micro-informatique.

★ Roux Seguela Cayzac & Goudard Atlantis, DCA, Lotus 1-2-3, Borland Paradox, IBM, 3 Com, Hewlett-Packard, AT&T, DEC, Ungermann-Bass, Olivetti, Nokia, SMT-Goupil sont des marques déposées

3616 INFO PC

Chaque jour toute l'info-micro au bout de vos doigts.

TOUTES LES CLES NE SE RESSEMBLENT PAS...

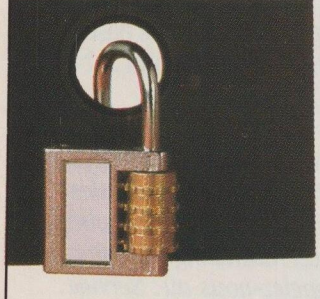
— Depuis 6 ans, MICROPHAR a vendu plus de 400000 clés à 2100 SSII et grandes entreprises. Ce succès atteste du sérieux et de la pérennité de nos prestations.

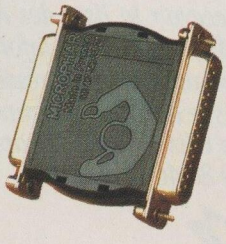
— Toutes nos clés possèdent un câblage interne personnalisé par client : le niveau de sécurité en est considérablement renforcé.

— La conception et la fabrication (composants CMS) sont intégralement réalisées par MICROPHAR afin d'offrir fiabilité et rapidité d'adaptation aux nouvelles machines.

— Une assistance technique structurée maintient en permanence notre système de protection dans plus de 55 langages de programmation sous DOS, XENIX, OS/2 et WINDOWS.

— Nos clés possèdent un haut niveau de compatibilité et sont disponibles dans huit couleurs différentes (avec marquage individualisé optionnel).





produits brevetés

Notre gamme de produits de ***protection de logiciels :***

— Une **clé électronique** contre le piratage

— Une **clé à mémoire** pour la protection sophistiquée, la location de progiciels, la protection de modules complémentaires et toute utilisation nécessitant un compteur (mémorisation de dates, mot de passe, etc.) :

- 31 mots de 16 bits disponibles en lecture et écriture
- 31 mots de 16 bits réservés au contrôle des opérations d'écriture
- Possibilité d'écriture (sans adaptateur), même chez l'utilisateur final

— La clé à mémoire est disponible sur MACINTOSH.
— Une clé à MICROPROCESSEUR pour micros, minis et toutes machines disposant d'un port série.

MICROPHAR, *leader européen des protections matérielles sur micro-ordinateurs, est distribué dans 11 pays d'Europe et d'Amérique. (Belgique : 091 21 11 17 / Suisse : 024 21 53 86).*

S.A. au capital de 1 800 000 F - 122, avenue Charles de Gaulle - 92200 Neuilly-sur-Seine - Tél. : (1) 47 38 21 21

Sommaire

- 12 -

L'ARCHITECTURE DES COMPILATEURS

De l'analyse syntaxique à la génération de code optimisé, les compilateurs obéissent pour la plupart à la même organisation.

- 18 -

LES PROCESSEURS PARALLÈLES

Le concept d'architecture parallèle des disques durs fait l'objet de nombreuses recherches fondamentales.

- 22 -

LA GESTION DE LA COULEUR

Une étude complète des techniques qui président à la création de l'image numérique couleur.

- 36 -

L'OBJET DANS LES BASES DE DONNÉES

La notion d'objet intègre peu à peu le système relationnel des gestionnaires de bases de données, via le C++ ou le langage Ada.

- 42 -

TURBO VISION

A la découverte de la bibliothèque objet du Turbo Pascal 6.0 : l'exemple pratique autour de la création d'un éditeur.

6 - RE-SOURCES
Une sélection de nouveaux outils de développement.

8 - A DÉVELOPPER
Les toutes dernières informations sur les normes, les langages et les systèmes.

58 - LES MAXIMES DU PROGRAMMEUR
Un agenda ponctué de quelques réflexions.

60 - LE CARNET D'ADRESSES DU DÉVELOPPEUR
Toutes les coordonnées des sociétés citées dans ce numéro 1 .

61 - L'ENQUÊTE LECTEUR -
Participez à l'évolution de Langages & Systèmes

64 - G.O.L.D.
Participez dès maintenant au Grand Open Logiciel et Développement.

Éditorial

La naissance d'une nouvelle publication est toujours un événement, pour la rédaction certes, mais avant tout pour les lecteurs, avides d'un autre type d'informations . En l'occurrence, les nombreuses enquêtes de lecture menées entre autres par le Laboratoire de Test du mensuel INFOPC font souvent apparaître la même interrogation: *"Comment trouver en France des informations précises et des articles suffisamment solides sur le monde du développement et la programmation de nos systèmes?"*. En clair, nos lecteurs révèlent qu'il n'existe rien au-delà d'INFOPC, qui puisse satisfaire les besoins des développeurs indépendants ou travaillant en SSII, et que l'angle hobbyiste est à écarter.

Et ce constat se trouve conforté par le succès du premier salon Software Dévelopment, qui s'il ne s'affiche pas résolument micro, traduit lui aussi des préoccupations de haut de gamme pour tout ce qui concerne la mise en œuvre d'applications professionnelles. La création de ce supplément bimestriel fut donc élaborée comme l'extension naturelle d'INFOPC, un complément à l'adresse d'un lectorat très exigeant. Autres choix, la rupture volontaire du rubriquage classique, à l'évidence inadapté à la nature des informations à dispenser, et une certaine latitude dans la pagination qui permet aux auteurs d'étudier les sujets sans limitation de volume. Un souhait aussi, que Langages & Systèmes serve de relai, de passerelle et de média privilégié à tous les développeurs : nos colonnes sont ouvertes à toutes les informations exclusives qui feraient avancer la communauté des programmeurs, avec pour seul mot d'ordre la valeur ajoutée.

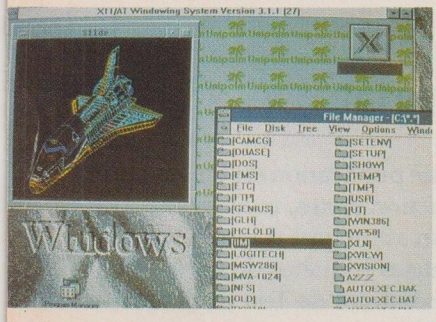
Satisfécit enfin, l'accord éditorial entre Langages & Systèmes et le Microsoft System Journal qui fait autorité dans son domaine. C'est grâce à ce partenariat rédactionnel hors-pair que les lecteurs découvriront dès le prochain numéro, les meilleurs articles de cette revue de réputation mondiale. De quoi sceller un contenu déjà riche, et répondre de la façon la plus efficace qui soit aux attentes d'un public pour l'heure frustré. Rendez-vous est d'ailleurs déjà pris par les lecteurs pour le numéro 2 de Langages & Systèmes, à la fin avril.

Emmanuel Alexandre

Re-sources

Le PC devient terminal

Le logiciel X11/AT, introduit en france par les Editions de la Boyère, est une application Windows 3 qui traduit de façon automatique tous les ordres X11 en ordres Microsoft Windows. C'est ainsi qu'il est possible de transformer un PC AT ou 386 sous Windows 3 en un véritable terminal graphique sous X-Windows Unix. Le X11/AT permet en outre d'employer toutes les capacités graphiques des cartes VGA ou haute résolution fonctionnant sous Windows 3, en monochrome, 16 ou 256 couleurs. Le logiciel est proposé au prix de 5 250 F HT.



PC et mini

Bachman Information System a développé Analyst Capture, qui permet d'extraire des données et des graphiques depuis les bases de données mainframe écrite en PL/1, IMS ou Cobol pour les importer dans OS/2 sur PC. L'administrateur de la base trouvera en outre un module expert dans le produit, qui étend la connaissance des règles de modélisation de base. Les utilisateurs micro peuvent ainsi analyser les graphiques et illustrations à la manière des systèmes d'information et d'aide à la décision (SIAD). D'après la compagnie, les experts comme les novices savent tirer parti d'Analyst Capture.

Une bibliothèque pour TP6

Innosoft lance simultanément avec Borland la première bibliothèque professionnelle compatible avec le Turbo Pascal 6. Object professional facilite par exemple le réemploi de modules déjà écrits et testés, et offre plus de 100 types d'objets : interfaces de type PM ou Windows, support souris, menus déroulants, générateurs d'écrans de saisie et de menus, gestion de fichier d'objets, impresions de formulaires, sous-programme pour créer des résidents aves swapping dans 6 Ko, etc.. Object professional possède également de nouvelles boîtes de dialogues de style SAA/CVA, support EMS 4.0. Tous les objets, les méthodes et routines sont livrés avec le code source complet, et les applications ne sont soumises à aucune royalties. L'éditeur fournit une documentation en français et le support technique gratuit.

ADA de A à Z

La convivialité et le fenêtrage gagnent désormais le langage Ada, par l'entremise de AdaZ qui se présente comme un environnement de développement professionnel complet intégrant logiquement les compilateurs Ada dans le monde des PC et PS sous DOS. Il inclut le compilateur, l'éditeur de texte, le gestionnaire de librairie, et le dévermineur symbolique. De plus, un ensemble de composants logiciels fourni avec l'environnement permet, depuis une application Ada, d'accéder aux ressources du système d'exploitation, de disposer d'une interface graphique très performante et d'effectuer des opérations de bas niveau.

AdaZ

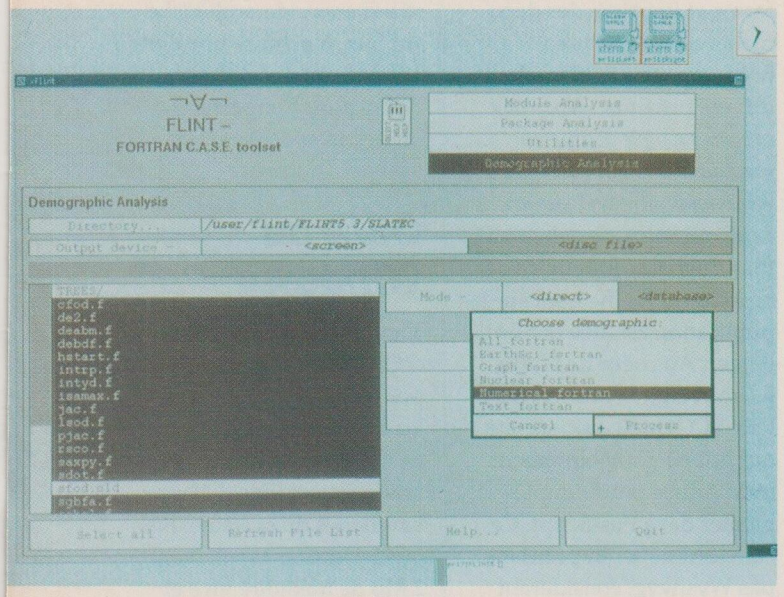
• Compilateur validé par l'Ada Joint • Programming office • Disponibilité des pragma Pack et Interface • Disponibilité des paquetages définis par le manuel de référence • Gestionnaire de tâches préemptif • Optimiseur d'analyse de code • Outil de recompilation • Gestionnaire de librairie • Interface graphique dotée de fonctions de gestion d'écran et de fenêtres, de tracé de figures géométriques et de dessin "Bit Map". • Un manuel d'utilisation, un manuel de prise en main, un manuel technique, un manuel pour l'interface Dos, un manuel de référence Ada.

Suite logique

Après avoir lancé l'intégré d'édition Ragtime 3 et le gestionnaire de documents Marco Polo, MacVonk complète sa ligne bureautique en introduisant P.INK SQL, une base de données compatible avec la ligne de produit précédente et disposant d'un langage de requête. Basé sur le modèle client/serveur, P.INK SQL est d'après les dirigeants de MacVonk la première base de données pour Macintosh qui tire parti des performances actuelles des stations de travail. Au travers de cette interface, les logiciels tels que 4D, Excel, Wingz ou FoxBase peuvent être utilisés comme applications Front End. P.INK SQL fontionne indifféremment sous Ethernet ou LocalTalk. En complément de cette base de données, MacVonk prépare un package développeurs incluant le serveur 5 postes, un lien avec Hypercard, le code source, une version Runtime et Campus, la version locale sur un Macintosh (Client et serveur).

Analyse de code

Ritme International vient de conclure avec la société anglaise Programming Research Limited un accord de distribution exclusif pour la France du logiciel QA For-



tran. Cet ensemble d'outils sophistiqués est avant tout destiné à l'analyse syntaxique et structurelle de programmes Fortran. QA analyse ainsi plus de 80 000 lignes de code par minute sur une station RISC. Il intègre également une interface X-Windows permettant une obtention simple et interactive des résultats. Au niveau structurel, QA produit une analyse globale du code, et intègre un module appelé Callpack qui vérifie la conformité des interfaces avec plus de 5 000 subroutines standards telles que GKS, PHIGS, et NAG. Un module d'analyse démographique permet de construire des mesures quantitatives objectives du code analysé en le comparant à une base de données construite à partir de plusieurs millions de lignes Fortran. QAC, l'équivalent pour l'analyse de code C, est annoncé pour ces prochaines semaines.

Objets graphiques

Microdata Soft annonce la nouvelle version de l'outil de génération automatique de code C++ et Turbo Pascal 5.5 à partir de graphiques Object Vision. L'éditeur graphique permet en outre de définir des interfaces permettant de tester les objets à l'aide du simulateur. Une fois un objet défini, il est possible de l'encapsuler pour une utilisation plus aisée, puis de générer automatiquement un programme complet en C++ ou en Turbo Pascal 5.5.

Un AGL graphique

L'outil de conception Silverrun (à l'origine sur Macintosh), est désormais disponible pour Windows3 et OS/2 PM. Cet atelier qui fonctionne selon la méthode Merise dispose d'un code unique et d'une interface identique quelle que soit la plate-forme de travail. Les fichiers sont de plus 100% compatibles entre les différents environnements. Les principaux outils qui composent Silverrun sont EAX, la modélisation conceptuelle de données, MRD la conception relationnelle, et DFD pour les diagrammes de flux de données. Dans les prochaines semaines, le dictionnaire interne de coordination (DIC) devrait voir le jour, en permettant aux membres d'une équipe de développement de partager des informations entre ces différents modules, afin d'assurer la cohérence des projets. Silverrun est distribué par ISE-Cegos à travers son département Environnements de développement. Chaque module est commercialisé au prix de 15 000 F HT. La station de travail conseillée commence avec un 386 doté de 4 Mo de mémoire vive (six pour la version PM) et de 4 Mo libres sur le disque.

Silverrun EAX

• Taille des modèles illimité • Contrôles du respect du formalisme, vérification des noms doubles à la création • Ouverture de quatre modèles en même temps • Taille des objets variable • Cadrage automatique des objets selon le contenu • Liens décomposables en segments • Liens tendus automatiquement • Mise en évidence des objets et des propriétés • Choix de l'affichage du modèle • Choix des fontes et des styles pour tous les libellés du modèle • Visualisation du modèle quelle que soit sa taille cohérence permanente graphique/dictionnaire • Rattachement des propriétés directement sur le modèle.

Silverrun MRD

• Récupération d'un modèle conceptuel de données créé avec l'EAX pour une transformation automatique en modèle relationnel. • Puissance graphique similaire au module EAX • Création de "champs utilisateur" • Prise en compte de toutes les spécificités en fonction du type de SGBD cible • Spécification des indentations primaires • Possibilité de collection et placement optionnel d'index physiques • Vérification d'intégrité et épuration du dictionnaire selon les besoins • Visualisation graphique du module relationnel • Génération automatique des noms d'index • Visualisation d'un fichier texte.

Silverrun DFD

• Capacités graphiques adaptées à la construction des diagrammes. • Sauvegarde automatique, mot de passe • Possibilité de fusionner et de scinder des diagrammes • Gestion des éclatements dans des fenêtres séparées • Gestion de la cohérence entre les niveaux • Récupération des objets utiles du niveau "père" au niveau "fils" lors d'un éclatement • Nombre d'objets illimités par diagramme • Choix des fontes et du style pour tous les libellés des programmes • Gestion de la documentation globale ou par diagramme • Diagnostic d'intégrité.

Librairie graphique

Média Sybernétics (représenté en France par les Editions de la Boyère) annonce une version professionnelle de la librairie graphique Halo, qui autorise dorénavant la réalisation de développements en mode protégé pour les langages Assembleur, C, Fortran, Ada, et en mode réel pour le Basic. Les Dos extender compatibles avec Halo Porfessional qui permettent de rompre la barrière des 640 K sont : Rational System Dos 16 M, Phar Lap, et Ergo. Le prix de Halo Professional est de 5 950F F HT pour tous les langages, hors Ada commercialisé à 7 950 F HT. Micro Sigma dispose par ailleurs de la nouvelle version de Case Windows, baptisée Corporate Edition, et qui entre autres améliorations supporte de multiples fenêtres enfants, est compatible MDI, et au choix SAA/CUA. Le logiciel est commercialisé au prix de 9 950 F HT.

A développer

Clipper: une version 5 très attendue

D'après notre confrère britannique PC Business World, Clipper continue de poser quelques problèmes aux développeurs faisant office de béta-testeurs. En l'occurrence, il semble que la gestion de la mémoire vacille dès lors que l'on utilise les fonctions C de la version 5, spécialement avec les librairies add-on de Clipper 87. D'après Dirk Lesko, le développeur américain auteur de la bibliothèque Funcky, le gestionnaire de mémoire virtuelle essaie parfois de ponctionner une partie de la mémoire provenant de la pile du système "d'objets virtuels" (VMM) pour l'ajouter à la pile mémoire contenant les variables statiques. La solution passe, entre autres, par XUNLOCK : dès qu'un programme en langage C en a terminé avec un paramètre, les développeurs doivent opérer un XUNLOCK sur ce dernier afin de libérer la mémoire qu'il occupe pour que le VMM puisse le déloger. Si XUNLOCK est effectivement incorporé dans la version 5 de Clipper, il n'est pas documenté, et les informations s'échangent officieusement. Une prochaine mise à jour du produit documentera cette fonction qui exigera d'autre part moins de mémoire conventionnelle, surtout dans un environnement de type réseau. Par ailleurs, les bibliothèques en C des développeurs permettront l'accès au VMM de Clipper 5 par le biais d'une interface de programmation en cours de réalisation. De par sa structure détaillée, cette API ne sera en revanche pas fournie automatiquement à tous les développeurs. Nantucket Clipper 5.0 apporte en outre un ensemble de nouveautés conceptuelles telles que les "commandes définies par l'utilisateur". Quoi qu'il en soit, PC Technologie diffuse cette version 5 depuis la mi-février.

Tout pour la couleur

Un comité de développeurs indépendants travaillant sur les produits orientés couleur vient de former un consortium dans le but de définir un protocole de gestion de la couleur standard pour les moniteurs et périphériques. Officieusement appelée "l'association des développeurs couleur", cette équipe a tenu son premier meeting lors de la MacWorld de San Francisco, et entend développer une API qu'Apple pourrait inclure dans la boîte à outils de son système d'exploitation. L'éventualité d'un portage de cette API dans le monde DOS et Windows est également envisagée. Dans les rangs de l'association, on trouve pêle-mêle Adobe, Aldus, Eastman Kodak, Hewlett-Packard, Microsoft, Radius, Supermac et Tektronix. A ce titre, outre les sociétés influentes sur Macintosh, on remarque la présence du leader des systèmes d'exploitation sur PC, et l'un des ténors de l'impression. Le but avoué de cette mise en commun des ressources est d'arriver à ce que l'utilisateur mette en forme et imprime des documents couleur sans se préoccuper de l'OS ou des drivers présents. «*Vous pouvez par exemple définir un graphique vert qui doit également apparaître vert sur une autre machine*», signale Jonathan Seybold, éditeur du rapport du même nom sur la publication asssitée par ordinateur. Au coeur de l'offre, se trouverait la norme couleur YCC de Kodak, société qui travaille conjointement avec Adobe pour l'implémenter dans PostScript. Ce standard ne serait pourtant pas Wysiwyg, considérant que les imprimantes ne supportent pas autant de couleurs que les moniteurs. Cependant, les utilisateurs pourront avoir un aperçu de leur sortie. Dans ce domaine, les avantages historiques d'Apple pour tout ce qui concerne le graphisme et l'édition électronique placent le Macintosh en tête de liste des préoccupations. «*Il n'est qu'à considérer le support moyen de la couleur par Windows pour en conclure que le portage sous DOS n'est pas seulement une question de stratégie*», conclut John Warnock, président d'Adobe.

Galaxie dBase

Ashton-Tate serait sur le point de présenter la béta-version de son compilateur dBase multi-plate-forme, qui serait prêt à 78%. Autre annonce, les versions de dBase IV pour les stations de travail Sun sous Unix 386. D'après l'un des testeurs, «*la vitesse est grandement accrue, et le code semble plus propre que dans la version DOS*». A titre d'essai, ce dernier a réussi en mode multitâche sur Sun, à extraire une donnée en temps réel depuis dBase pour l'importer dans un dessin sous Autocad.

OS/2 ou Windows

Tandis que les adeptes d'OS/2 se félicitent de la disponibilité de la version 1.3, les sceptiques affirment qu'il ne s'agit pas là d'une amélioration susceptible d'élargir l'audience du système. Bien que nécessitant seulement 2 Mo de mémoire vive et disposant d'une vitesse de traitement en progrès substantiel, les bastions hors du giron IBM n'ont pas témoigné plus d'engouement que par le passé. Comme le précise le responsable système d'une raffinerie en Californie «*si vous tirez avantage d'un fonctionnement en multitâche, OS/2 vous concerne, dans le cas contraire, la version 1.3 ne vous fera pas changer d'opinion*». Si le gain de vitesse est appréciable, d'aucuns notent que cette performance accrue perd de son efficacité pour le cas d'un fonctionnement en multitâche dans seulement 2 Mo de mémoire. A l'inverse, OS/2 se montre plus véloce que Windows 3 dans cette configuration mémoire, puisqu'il sait mieux gérer cette dernière. Peu de gens pensent que la version 1.3 d'OS/2 pourrait déstabiliser Windows 3, dont la nouvelle version est annoncée "solide et sans bugs majeurs". Un rapport récent du Forrester Group précise d'ailleurs que d'ici la fin de 1991, 40% du parc installé pourrait confier ses applications à Windows.

DOS ◆ POWER ◆ TOOLS

Enfin disponible en français pour 378 F. seulement !*

EN CADEAU**

MS-DOS
Micro-Référence

* *Disquette incluse.*

** Offre valable jusqu'au 15/04/91 dans la mesure des stocks disponibles

UN REGARD NOUVEAU SUR LE DOS

Vous cherchez à optimiser votre DOS. Vous souhaitez gagner du temps. Vous le pourrez enfin, grâce à cet ouvrage de plus de 1000 pages qui fait aujourd'hui autorité. Des centaines d'astuces et utilitaires provenant des colonnes de l'édition américaine de PC Magazine, largement complétés et enrichis par les plus grands experts, vous feront découvrir le DOS dans ses moindres détails. Comme des milliers de lecteurs avant vous, faites confiance à cet outil d'exception !

Cet ouvrage contient plus de 295 programmes assembleur prêts à l'emploi, plus de 60 nouveaux utilitaires puissants, des éléments performants sur la gestion de la mémoire des nouveaux ordinateurs de type 386, sans oublier toutes les évolutions du DOS : la version 4.0 et le DOS Shell.

En vente chez votre revendeur habituel.

BON DE COMMANDE

à renvoyer à SYBEX,
6/8 Impasse du Curé,
75882 Paris Cedex 18

Société _____

Nom _____

Prénom _____

Fonction _____

Rue _____

Ville _____

Code Postal |_|_|_|_|_|

SYBEX

Oui, je désire commander :
_____ *exemplaire(s) de* ***Dos Power Tools*** *au prix de* ***378 F*** *Franco TTC l'unité et recevoir mon cadeau, le livre MS-DOS Micro-Référence.*
N'oubliez pas de spécifier le format de disquette souhaité :
☐ *3" 1/2* ☐ *5" 1/4*
☐ *Je règle par chèque bancaire ou postal la somme de* _____ F
☐ *Je règle avec ma carte bleue la somme de* _____ F
N° de ma carte |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
Date d'expiration |_|_| |_|_|
Date *Signature*

TÉL. : (1) 42.09.95.95
TÉLÉCOPIEUR : (1) 42.09.01.45

☐ *N'oubliez pas de demander notre nouveau catalogue général 1991.*

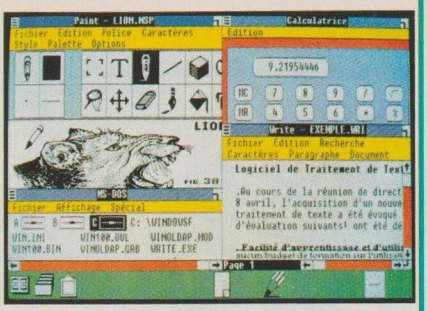
TOUT POUR WINDOWS 3.0

TOUS LES DEVELOPPEURS

- *Plus de 1 500 produits de développement présentés dans le catalogue le plus complet du marché.*
- *La garantie des prix les plus bas : nous nous alignons sur tout prix de concurrent publié le même mois*
- *Un stock important pour vous livrer rapidement*
- *Des spécialistes prêts à vous écouter et à vous conseiller*

WINDOWS 3.0, VF

1390 FHT 1648 FTTC



	Prix HT	Prix TTC
WINDOWS 3.0 Development Toolkit	3690	4376

Tous les outils nécessaires au développement d'applications-Windows Multitâches. Déboggueur Codeview adapté. Accompagné de C ou Pascal, l'environnement de programmation est complet. Version US. Techno-Fax n° 320271

WINDOWS 3.0 GUIDE Frantz, VF 339 356

Probablement, le meilleur livre de développement jamais écrit à ce jour sur Windows 3.0.

TOOLBOOK, Asymetrix, VA 3390 4020

Toolbook permet de développer une application sous Windows 3.0 très rapidement (sans compilateur de SDK), grâce à des outils prédéfinis : boutons, objets graphiques, champs en format texte... Techno-Fax n° 125116

ACTOR 3.0, Whitewater, VA 6990 8290

Environnement de programmation objets pour Windows. Liens dynamiques avec le C, Pascal, Assembleur. Déboggueur, browser et éditeurs intégrés. Gestion de la mémoire virtuelle. Techno-Fax n° 320344

TWINTALK, Sonotec, VF 1990 2360

Gére vos modems (de 75 à 19200 bauds) compatibles HAYES sousWindows. Protocoles Kermit, XModem, YModem, PDip. Emulation VT102. Techno-Fax n° 125117

WIN JT Base, JT Diffusion, VF 2360 2799

Gestionnaire de fichiers de type dBase III+ sous Windows. Editeur intégré. Langage de requête propre, générateur de rapports et d'étiquettes. Techno-Fax n° 125119

COMMONVIEW 2.0, Glockenspiel, VA

Inclut Commonview C++. Il permet de développer des objets paramétrables et de simplifier les développements sous Windows 3.0. Nécessite C 6.0 et Windows SDK. Version US. Techno-Fax 320191

TOUT POUR dBASE / FOX / CLIPPER

	Prix HT	Prix TTC
CLIPPER 5.0, VF	7590	9002
CLIPPER 5.0, VA	5490	6511
dBASE III+, Ashton Tate, VF	5480	6499
dBASE IV, Ashton Tate, VF*	6350	7531
dBFAST PLUS, VA	2695	3196
dBXL, WordTECH*	2990	3546
FOXBASE 2.1+, Fox Software, VA	2990	3546
FOXPRO, Fox Software, VF	6710	7958
PARADOX 3.0, Borland, VF	5650	6701
DATAEASE 4.0, VF	6190	7341

GENERATEURS D'APPLICATIONS

dBOUTILS III+, LCE (5¨), VF	590	700
dANALYST, INNOSOFT, VF	2650	3143
GENIFER, ByFel, VA	2690	3190
SYCERO, System C, VF	5390	6393

BIBLIOTHEQUES

dGE 3.0, VF	2390	2835
R & R, VA	1490	1767
dBU, INNOSOFT	2690	3190
dBCOM, VF	2690	3190
NANTUCKET TOOLS, VA	3790	4495
SILVER PAINT, VA	1090	1293
FUNCKY, dLESKO, VA	1990	2360
CLEAR FOR dBASE	1590	1886
CLIPPER API TOOLKIT, Deskwiew, VA	1490	1767

SOURCER : UN VÉRITABLE DÉSASSEMBLEUR

SOURCER est un désassembleur très performant. Le code généré est clairement commenté, utilisable pour le réassemblage et ressemble au code source original. SOURCER détermine même les directives assembleur adaptées aux programmes multisegments. Accepte les fichiers COM, EXE, SYS, BIN. BIOS PRE PROCESSOR associé avec SOURCER permet d'obtenir des listings de code source de n'importe quel BIOS UNPACKER, "dépaquette" les .EXE et identifie les programmes Overlays rattachés.

	Prix HT	Prix TTC
ASMTOOLS analyse le code source.		
SOURCER	990 F	1174 F
SOURCER AVEC BIOS PREPROCESSOR	1195 F	1417 F
UNPACKER	490 F	581 F
ASM TOOLS	1490 F	1767 F

TECHNO-FAX : 320 340

C- COMPILATEURS

	Prix HT	Prix TTC
MICROSOFT C P D S 6.0, VF	3090	3665
MICROSOFT QUICK C , VF (Promo = 690 F)	1090	1292
LATTICE C, 6.0, VA	1690	2040
TOPSPEED C, Jensen & Partners, VA	1690	2004
NOVELL C/386, Novell, VA	8490	10069

C- BIBLIOTHEQUES ECRANS

HIGH SCREEN 5, PC SOFT, VF	3950	4685
C SCAPE 3.2, Oakland Group, VA	3990	4732
Greenleaf Datawindows, VA	2890	3427
Metawindows T/C, Metagraphics	990	1174
Power Screen, Blaise, VA	1090	1293
Vermont Views, Vermont Creative, 20,VA	4390	5206
Vitamin C, Creative Programming, VA	1790	2123

BRIEF 3.0

- Un langage macro puissant adaptable à vos besoins. Vous ne manquerez pas d'apprécier ses touches de fonctions programmables, son déboggueur, le code source intégré... Adapté à 36 compilateurs (Ada, C, Pascal...) avec module dBrief pour langage dBase.

Capacité de 4 milliards de lignes par fichier. *Techno - Fax n° 320323*

Version Française/Dos :	3250 FHT	2150 TTC
Version US :	1690 FHT	2549,90 TTC
Version OS/2 - US :	1690 FHT	2004,34 TTC
dBrief US :	950 FHT	2004,34 TTC

TECHNO - FAX : 320 323

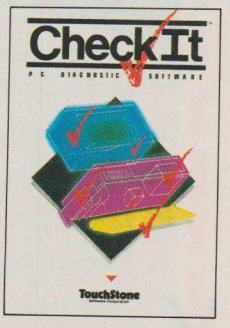
C- BASE DE DONNEES

	Prix HT	Prix TTC
BTrieve, Multi-utilisateur, Novell, VA*	4490	5325
C-ISAM, Informix, VA	1990	2360
Code Base 4, INNOSOFT, VF	2595	3078
Hyperfile, PC SOFT, VF (5¨)	3950	4685

CHECK IT :

Trouvez la solution à vos problèmes !

CHECK IT effectue automatiquement un diagnostic de votre configuration et vous aide à trouver la solution à votre problème grâce à la description de la configuration (interruption, table CMOS) outils intégrés (mémoire, ports), banc d'essai (calcul vitesse, affichage) boite à outils pour modifier ou réparer. Indispensable. Version US. Techno-Fax n° 400142



~~1500 FHT~~ 990 FHT 1174,14 FTTC

LOGICIELS PC SOFT

HIGH SCREEN 5 *(TECHNO-FAX : 320 288)*

Générateur d'écrans très puissant (texte, graphique, clavier, souris) compatible multilangage, VF en 5¨1/4

prix HT prix TTC

~~4900~~ 3950 4684,70

HYPERPRINT 2

Pour dessiner vos rapports à l'écran et imprimer jusqu'à 9 fichiers liés, VF En 5¨1/4

~~4900~~ 3950 4684,70

HYPERFILE

Séquentiel indexé pour la gestion des fichiers dans le langage de votre choix, VF En 5¨1/4

~~4900~~ 3950 4684,70

HYPERPACK

Comprend les 3 produits précédents en 5¨1/4

~~9700~~ 7900 4684,70

C- BIBLIOTHEQUES GENERALES/GRAPHIQUES

	Prix HT	Prix TTC
C Asynch Manager, Blaise, VA	1390	1648
C Function Library, Sterling Castle, VA	990	1174
C Tools Plus, Blaise, VA	1090	1292
Greenleaf Superfunctions, VA	2190	2597
Essential Graphics, Essential Software	3290	3902

B- COMPILATEURS BASIC

COMPILATEUR BASIC 4.5, Microsoft, VF	790	937
TURBO BASIC 1.1, Borland,VF	790	937
MICROSOFT BASIC PDS 7.1, VF	2990	3546

B- BIBLIOTHEQUES BASIC

DBLIB, AJS, VA	1290	1530
Turbo Screen, PC Soft, VF	990	1174
QUICK FILE, Somma, VF	710	842
QUICK PAK PRO, Crescent, VA	1290	1530

E- EDITEURS

SPF / PC, Command Technology, VA	2150	2550
Sage Professional Editor, Sage, VA (DOS)	1790	2122
KEDIT, Mansfield, VA	1290	1530
Norton Editor, P. Norton, VA	530	629

F- FORTRAN

FORTRAN 5.0, Microsoft, VA	3490	4139
Lahey Personal	890	1055
Printmatic, Microcompatibles, VA	1250	1482
MATHLIB, Wiley, VA	1990	2360

G- OUTILS GRAPHIQUES

BABY DRIVER 1790 FHT 2123 FTTC

Bibliothèque avec possibilité d'impression d'écrans, ou d'images BITMAP en mémoire. Plus de 175 imprimantes supportées. Compatible C, TC ou C++. Version US. Techno-Fax n°320215.

PCX Programmer's Toolkit 2100 FHT 2495FTTC

Permet d'inclure des dessins de type PCX dans toutes vos applications. Multilangage, ce toolkit offre plus de 75 fonctions différentes. Supporte la mémoire EMS 4.0. Version US.Techno-Fax n°320187

HALO 88 2790 FHT 3310 FTTC

Plus de 220 fonctions graphiques : manipulation d'images, fenêtrages, compression d'images, gestion de cartes graphiques, contrôle du curseur, gestion des imprimantes ou scanners. Compilateur spécifique (C, Fortran, Pascal, Basic). Version US. Techno-Fax n° 320242

METAWINDOW 1695 FHT 2010 FTTC

Outil de développement d'interfaces graphiques avec de nombreuses primitives. Plusieurs cartes graphiques supportées. Pas de royalties. Version US compatible C ou Pascal. Techno-Fax n° 320247

L- LINKERS

BLINKER, VA	1890	2241
PLINK 86 +, Sage, VA	3590	4258
RT LINK/Plus 4.10, VA	3190	3783

P- COMPILATEURS PASCAL

PASCAL COMPILER Innosoft, MICROSOFT, VA	2990	3546
QUICK PASCAL, Microsoft, VF (Promo 690 F)	1090	1293
TURBO PASCAL V5.5, Borland, VF	1090	1293
TURBO PASCAL PRO V5.5, Borland, VF	2195	2603

* Promotions valable jusqu'au 31/3/1991 ; VF = Version Française - VA = Version Anglaise

VONT CHEZ TECHNO-DIRECT

LE NOUVEAU CATALOGUE MAC N°2 EST ARRIVÉ

• Un service "TECHNO - FAX" inédit vous permet d'obtenir gratuitement à partir de votre télécopieur toute documentation logicielle de votre choix au (1) 45.06.67.01

• Un service "FOUINEUR" gratuit vous confirme en moins de 48 heures, prix et délais de livraison de n'importe quel logiciel ou matériel que vous recherchez aux USA.

DEVELOPPEZ EN TURBO-PASCAL 6.0

OBJECT PROFESSIONAL
La bibliothèque professionnelle pour la programmation orientée objets, adaptée au 6.0. Plus de 100classes objets pour développer votre productivité : fenêtrage,menus, saisie de données... Boites de dialogue SAA/CUA, supportXUS/EMS. Code source inclus. V.F

Techno-Fax n° 320110 ~~2490 FHT~~ **2095 FHT** **2484,67 FTTC**

TSR MADE EASY
Vous aide à transformer vos programmesTurbo-Pascal en programmes résidents. Code source inclus. VF

~~890 FHT~~ **750 FHT** **889,50 FTTC**

TURBO BTREE
Vous permet d'écrire vos applications de bases de données. Deux milliards d'enregistrements, 100 clés par fichier index,support réseau.¡VF

Techno-Fax n° 320108 ~~1795 FHT~~ **1590 FHT** **889,50 FTTC**

P- BIBLIOTHEQUES PASCAL

	PRIX HT	PRIX TTC
DATABOSS, Atea, version 3.0 VF	4795	5687
Asynch Plus, Blaise, VA*	1190	1411
POWER SCREEN, Blaise, VA	1090	1293
POWER TOOLS PLUS, Blaise, VA	1090	1293
Turbo-Analyst, Innosoft, VF	1095	1299
SCIENCE & Engineering Tools, Quinn curtis, VA	995	1180

O-OS/2

OS/2 V1.2, IBM, VF	2390	2834
OS/2 Presentation Manager Toolkit, VA	3950	4685
Greenleaf Datawindows OS/2, VA	3290	3901
Vitamin C OS/2, VA	2840	3368
Smalltalk/V PM	3990	4732

D- OUTILS DE DEMO

Dan Bricklin'S Demo II, VA	1490	1767
Grasp, Mace, VA	1150	1364
Instant replay Pro III, Nostradamus, VA	1190	1411
Show Partner F/X, VA	2890	3427

U- UNIX/XENIX

386/IX, Interactive, VA (Runtime ill.)	6490	7697
386/IX-TCP/IP, Interactive, VA	3600	4270
UNIX SYSTEMS V, SCO, Runtime, 2 ut., VA	5120	6072
VP/IX, 2 util., Interactive, VA	3690	4376
X-WINDOWS 386, ill., Interactive, VA	4720	5598

U- UTILITAIRES DOS

386 MAX PRO, Qualitas, VA*	1090	1293
Copywrite, Quaid, VA	780	925
Option Board de Luxe, VA	1490	4732
DR DOS 5.0, VF	840	996
LAPLINK III, VA	1090	1293
NORTON ADV, P. Norton, version 5.0 VA*	1290	1530
PC TOOLS 6.0, VF	1190	1411
QRAM, Quaterdeck, VA*	630	747
VIRUSAFE, Elashin, VF	590	700

LES MEILLEURS OUTILS C POUR 1991

MICROSOFT C PDS 6.0 ..... 3090 FHT 3665 FTTC
L'environnement idéal pour développer des applications sophistiquées en C. Compatible DOS, Windows, OS/2, PM, nombreux outils intégrés dans le nouveau PWB (Programmer's Workbench), éditeur puissant, compilateur, linker, utilitaire NMAKE, nouvelle version 3.0 du débogueur Codeview et Source Browser. *Techno-Fax n° 320260*

CODE BASE 4 ..... 2595 FHT 3078 FTTC
Bibliothèque C-ISAM compatible dBase et/ou Nantucket. Nombreuses fonctions intégrées (menus, fenêtrages, browser...) permettant d'accroître votre productivité. Compatible C, TC, OS/2 ou Windows, V.F
Techno-Fax n° 320212

GREENLEAF COMMLIB 3.1 ..... 2190 FHT 2597 FTTC
La meilleure bibliothèque de communication. Plus de 100 fonctions (XModem, YModem, Kermit, Xon/Xoff) avec contrôle des modems HAYES. Supporte les cartes multivoies et VARTS 16550 ! Code source intégré. Version US (précisez le compilateur). *Techno-Fax n° 320193*

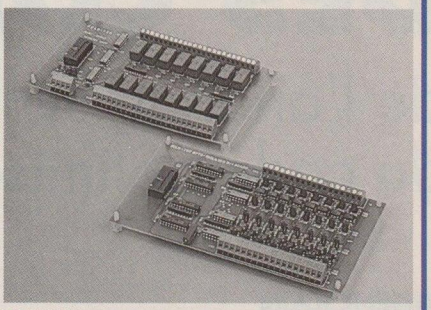
VITAMIN C ..... 1790 FHT 2123 FTTC
Permet de créer des interfaces utilisateurs exceptionnels avec menus, fenêtres, écrans de saisie... Plusieurs centaines de fonctions s'offrent à vous pour créer des fenêtres superposables, aide en ligne, fenêtres de dialogue, menus, support souris... Version US sous DOS, OS/2 et UNIX-disponible. *Techno-Fax n° 320152*

C++

Les nouveaux langages C++ comportent maintenant de nombreuses fonctions objets aux normes ATT 2 et ANSI C.

	prix HT	prix TTC
BORLAND		
Turbo C++, VF	1090	1293
Turbo C++ Professional, VF	2295	2722
ZORTECH		
C++, 2.1, VA	1290	1530
C++ Sources, VA	990	1174
C++ Tools, VA	990	1174
C++ Debugger, VA	990	1174
C++ Version Development	2715	1174
C++, ext os/2	990	1174
GLOCKENSPIEL		
C++ 2.0	5900	6997
Commonview 2/Zortech	3595	4264

PRODUITS SCIENTIFIQUES/CAO



LOGICIELS	prix HT	prix TTC
AUTOCAD VIO, VF	27500	32 615
CROSS ASSEMBLEUR, 2500AD, VA	2190	à 4 590
GRAPH IN THE BOX, ANALYTIC, VA	1590	1 886
MATHCAD, VF, Traitement de texte	3990	4 732
STATGRAPHICS, STSC,	7900	9369
TKSOLVER PLUS, UTS, VA	3690	4 376
Mathematica 386, Wolfram, VA	9890	11 730
Mathword, Apsylog, VF	2790	3 309

CARTES		
PCL 848A, Carte IEEE 488 pour PC/AT	3100	3677
PCL 720, Carte 32 entrées et sorties TTL, 3 compteurs	1700	2016
PCL 725, Carte 8 entrées opto-couplées, 8 sorties relais 220V/1A	2135	2532
PCL 422 XT, Carte RS 422 et RS 485, 2 voies COM1 et COM2 pour PC/XT	1525	1809
PCL 714, Carte A/D, D/A 14 bits et I/O TTL 16 entrées et 2 sorties analogiques 16 entrées et 16 sorties TTL, 3 compteurs	5025	5960

Catalogue complet des cartes scientifiques disponible sur simple demande.

TECHNO-DIRECT
(1) 47 28 62 90

FAX : (1) 47 28 62 89
TÉLEX : MBI 290266

- **Recevez gratuitement le catalogue de logiciels et matériels le plus complet du marché PC ou MAC.**
  (Ces pages ne sont qu'un extrait limité du catalogue)
- **Recevez gratuitement des documentations complémentaires sur les produits mentionnés dans ces pages.**

Retournez ce coupon-réponse à :
TECHNO-DIRECT
6, Bd Henri Sellier, 92150 Suresnes

- ☐ Je désire recevoir votre catalogue PC
- ☐ Je désire recevoir votre catalogue MAC
- ☐ Votre catalogue cartes scientifiques
- ☐ Votre futur catalogue windows.

NOM : ..... FONCTION : .....
SOCIÉTÉ : .....
ADRESSE : ..... VILLE : .....
C.P. .....
TEL. : .....

CONDITIONS D'ACHAT :
Horaires d'ouverture : 9 h - 19 h du lundi au vendredi.
CB acceptées contre R avec supplément.
Possibilité d'enlèvement sur place : Immeuble Eiffel, 6, rue des Bourets 92150 Suresnes
Prix donnés pour départ Suresnes, paiement comptant.
Frais de port France métropolitaine 55F (65,23TTC) pour toute commande inférieure à 1 500 F ; 150 FHT pour toute commande comprise entre 1 500 F et 5 000 F. Nous téléphoner pour les commandes supérieures et pour toute expédition à l'étranger. Le matériel livré est assuré par nos soins. Prix indicatifs modifiables sans préavis. Nous nous alignons sur tout prix fermes donnés par des concurrents.

L'ARCHITECTURE DES COMPILATEURS :

de l'analyse syntaxique à la génération du code optimisé

EUGÈNE KAIN

L'importance des compilateurs en tant qu'outils de production des développeurs de logiciels a suscité une recherche considérable afin de découvrir les manières les plus efficaces de les programmer et de faciliter leur implémentation, quel que soit le langage source et quelle que soit la machine de destination.
Cette recherche a permis de mettre en évidence neuf composants essentiels, que l'on retrouve dans la plupart des compilateurs actuels :

1 l'analyseur lexical (scanner).
2 l'analyseur syntaxique (parser).
3 l'analyseur sémantique (semantic analyzer).
4 le générateur de code intermédiaire (intermediate code generator).
5 l'optimiseur de code (code optimizer).
6 le générateur de code final (object code generator).
7 l'éditeur de liens (linker).
8 le gestionnaire de la table des symboles (symbol table manager).
9 le gestionnaire d'erreurs (error handler).

Il faut noter que cette décomposition d'un compilateur en 9 éléments est relativement arbitraire et peut varier d'un programme à l'autre. En effet, tous les compilateurs ne distinguent pas les diverses opérations aussi finement : il arrive par exemple que certaines fonctions de l'analyseur lexical soient intégrées dans l'analyseur syntaxique; réciproquement, certains compilateurs disposent de plusieurs phases d'optimisation : une première optimisation globale a lieu après l'analyse sémantique, puis une deuxième (appelée «optimisation locale») est effectuée après le générateur de code final.
Cependant, il est conceptuellement plus facile de conserver la décomposition en 9 éléments qui est présentée ici, tout en gardant à l'esprit le fait que l'implémentation réelle de ces 9 fonctions théoriques dépend du compilateur considéré. L'analyse lexicale d'un programme consiste à convertir le code source tel que le programmeur l'a écrit en un format interne mieux adapté à la suite des opérations. De manière plus précise, l'analyseur lexical lit le code source caractère par caractère et renvoie en sortie des unités de base lexicales, appelées des tokens.

Code source Pascal

C'est ensuite sur ces tokens que l'analyseur syntaxique sera amené à travailler. A titre d'exemple, considérons le fragment de code source PASCAL suivant:

```
Program Exemple;
{ Exemple de code source }
var x : integer;
begin
x := 25 - 2;
end.
```

L'analyseur lexical le découpera en tokens selon le modèle suivant, où les tokens sont séparés par le caractère '|' :

```
Program | Exemple | ; | var
| x | : | integer | ; |
begin | x | := | 25 | - | 2
| ; | end | . |
```

La sortie de l'analyseur lexical comportera donc les éléments chargés de sens du code source, groupés selon leur signification. L'intérêt de cette première étape est de normaliser la présentation du code source, afin de simplifier la phase suivante d'analyse syntaxique.
De manière plus précise, l'analyseur lexical doit effectuer les opérations suivantes :
- Identifier les unités de base lexicales du programme source (les tokens).
- Filtrer tous les caractères de séparation qui n'ont aucune signification lexicale dans le langage utilisé, par exemple les espaces redondants ou les retours chariot (ces caractères de séparation sont appelés whitespace).
- Filtrer les commentaires, puisqu'ils n'influencent pas l'analyse du programme.
- Détecter et indiquer les erreurs éventuelles : commentaires imbriqués, guillemets mal appariés, etc.

Rôle de l'analyseur syntaxique

Une fois ces opérations effectuées, l'analyseur syntaxique reçoit donc en entrée les éléments significatifs du code source, déjà regroupés en tokens. Son rôle est de vérifier que la syntaxe du code source satisfait bien aux règles

Chaque éditeur de compilateur vante la supériorité de son produit sur les concurrents. Mais au-delà des annonces publicitaires, la plupart des compilateurs partagent la même architecture de base. Comprendre les mécanismes fondamentaux de la compilation constitue donc le premier pas vers un choix en connaissance de cause.

grammaticales du langage considéré. Pour ce faire, l'analyseur syntaxique construit une représentation de la syntaxe du programme analysé, généralement exprimée sous la forme d'un arbre syntaxique.
Cette analyse est similaire à celle que l'on pourrait faire sur une phrase de français afin d'identifier ses éléments grammaticaux essentiels (sujet, verbe, complément, adjectif...).
L'arbre syntaxique construit pour une ligne de code source peut cependant avoir de nombreux niveaux, particulièrement lors de l'emploi d'expressions parenthésées. Voici à titre d'exemple une ligne de code source PASCAL suivie de son analyse syntaxique :

```
position := (vitesse *
temps) + 30;
ligne  <= ident1 ':='
express1;
ident1 <= 'position';
express1 <= express2 '+'
nombre;
express2 <= ident2 '*'
ident3;
ident2 <= 'vitesse';
ident3 <= 'temps';
nombre <= '30';
```

L'analyseur sémantique se situe, quant à lui, à un niveau de «compréhension» du code source très supérieur aux deux premiers composants étudiés jusqu'ici. En effet, les aspects les plus fins (et souvent les plus importants) de la plupart des langages ne sont généralement pas contenus dans la simple grammaire, mais dans des règles qui définissent les expressions légales ou non en fonction du contexte global du programme. Un exemple d'une telle règle concerne le type des variables dans un langage fortement typé comme PASCAL.
Considérons l'exemple suivant :

```
var x, y : integer;
x := 3; y := 3.5;
```

Ces lignes sont syntaxiquement correctes car elles satisfont à la grammaire du langage. Cependant la troisième ligne sera déclarée illégale par l'analyseur sémantique, à cause de l'incompatibilité de type entre les deux membres de l'assignation (celui de droite représente une expression de type «real», alors que la variable à gauche de l'opérateur d'assignation est de type «integer»).

Illégalité sémantique

D'autres exemples courants de lignes syntaxiquement correctes mais sémantiquement illégales concernent la redéclaration d'une variable ou d'une fonction, l'appel d'une fonction avec un nombre ou un type d'arguments différents de sa déclaration, ou encore l'utilisation d'une variable non déclarée.
Il apparaît donc que l'analyseur sémantique doit collecter des informations disséminées tout au long du code source pour être à même de remplir sa mission. Pour ce faire, ce module utilise le gestionnaire de la table des symboles auquel il fournit toutes les informations recueillies sur les variables et fonctions au fur et à mesure de leurs déclarations (en particulier le type des variables, le type et le nombre d'arguments des fonctions, etc.).
Une fois le programme source entièrement analysé et validé, la table des symboles contient normalement toutes les informations dont aura besoin le générateur de code intermédiaire. Ce générateur de code construit une représentation du programme source à partir des analyses précédentes. Cette représentation peut être conçue comme un programme destiné à une machine abstraite, et elle doit répondre à deux exigences essentielles : elle doit d'une part être facile à produire à partir des analyses syntaxique et sémantique, et d'autre part être facile à traduire dans le langage objet final (ou langage machine). Ce code intermédiaire peut prendre des formes très diverses selon les langages et les compilateurs; dans certains cas, il peut même s'agir directement du code objet définitif (mais avant optimisation).
Voici en exemple une ligne de source suivie du code intermédiaire qu'elle pourrait engendrer :

```
x := a + b * ( c + d );
temp1 := c + d;
temp2 := b * temp1;
temp3 := a + temp2;
x := temp3;
```

L'optimiseur de code a pour but d'améliorer le code intermédiaire dans deux directions principales : diminuer sa taille d'une part, et augmenter sa vitesse d'exécution d'autre part. Il faut

savoir que, dans certains cas, ces deux objectifs sont antagonistes.
C'est alors au programmeur d'indiquer au compilateur le type d'optimisation qu'il souhaite favoriser. Diverses techniques d'optimisation ont été développées pour répondre à la plupart des besoins courants dans ce domaine, mais il faut savoir qu'une optimisation poussée impose une analyse approfondie du code source, et ralentit donc sensiblement la phase de compilation.
A partir du code intermédiaire optimisé, il faut maintenant générer les instructions correspondantes dans le lan-

QUELQUES TECHNIQUES «CLASSIQUES» D'OPTIMISATION

■ Dans un monde idéal, les compilateurs produiraient un code objet aussi compact et rapide que pourrait l'écrire à la main un bon programmeur. Malheureusement, cet idéal est rarement atteint sur l'ensemble d'un programme. Toutefois certaines techniques existent afin d'améliorer le code intermédiaire généré par les algorithmes classiques, l'application de ces techniques permettant dans certains cas d'obtenir un code objet réellement optimal pour certaines portions de code source, et plus généralement d'améliorer la performance globale du programme objet (il semble que ce soit la vitesse qui constitue le but recherché de nos jours, bien plus que les économies de mémoire). Pour être intéressante, une technique d'optimisation doit satisfaire à trois critères :
- Préserver le comportement du programme source.
- Améliorer la vitesse d'exécution de façon mesurable.
- Présenter un bon rapport entre la difficulté d'implémentation et le résultat obtenu.
Etudions maintenant sur un exemple de code intermédiaire quelques optimisations couramment employées par la plupart des compilateurs :

```
t6 := 4*i;
a[ t6 ] := t3;
x := a[ t6 ];
t7 := 4*i;
t8 := 4*j;
t9 := a[ t8 ]; a[ t7 ] := a[
t9 ];
t10 := 4*j; a[ t10 ] := x;
```

Une première optimisation possible consiste à éliminer les calculs redondants qui interviennent à plusieurs endroits (common subexpression elimination). Dans l'exemple étudié, les assignations à "t7" et "t10" contiennent les mêmes expressions que les calculs de "t6" et "t8" respectivement. L'élimination de ces calculs redondants donne donc :

```
t6 := 4*i;
a[ t6 ] := t3;
x := a[ t6 ];
t8 := 4*j;
t9 := a[ t8 ];
a[ t6 ] := a[ t9 ];
a[ t8 ] := x;
```

Une autre optimisation possible s'appelle la propagation des copies (copy propagation). Il s'agit après une assignation du type "a := b" de remplacer toutes les références à "a" par des références à "b". Ici, ce cas de figure concerne les variables x, a[t6], et t3 :

```
t6 := 4*i;
a[ t6 ] := t3;
x := t3;
t8 := 4*j;
t9 := a[ t8 ];
a[ t6 ] := a[ t9 ];
a[ t8 ] := t3;
```

Les deux techniques présentées ci-dessus conduisent souvent à rendre certaines assignations complètement inutiles, car le contenu de la variable correspondante est changé avant qu'elle soit référencée. C'est le cas ici pour l'assignation de "t3" à "a[t6]", qui est ensuite suivie d'une nouvelle assignation à "a[t6]".
Dans ces conditions, la première assignation constitue du code inutile que l'on peut éliminer (dead code elimination) :

```
t6 := 4*i;
x := t3;
t8 := 4*j;
t9 := a[ t8 ];
a[ t6 ] := a[ t9 ];
a[ t8 ] := t3;
```

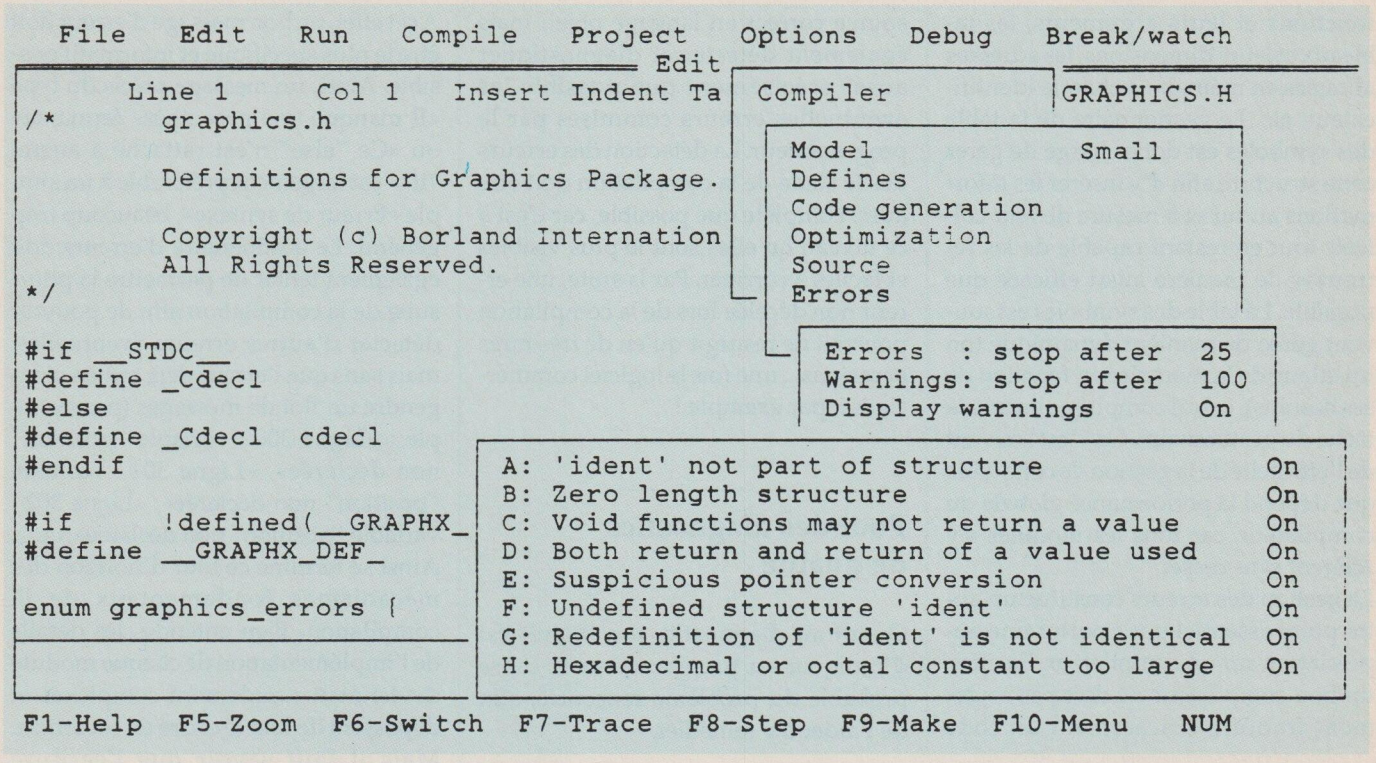
Si les méthodes ci-dessus permettent effectivement d'améliorer les suites d'instructions engendrées par le générateur de code intermédiaire, il existe cependant un autre cas particulièrement important où l'amélioration de performance peut être sensible : il s'agit du calcul d'une expression invariante à l'intérieur d'une boucle.
Des gains de temps substantiels sont réalisables en déplaçant le calcul invariant hors de la boucle (invariant code motion). On cherche ainsi à éviter d'effectuer les mêmes calculs à chaque itération, comme l'illustre l'exemple ci-après :
avant optimisation :

```
i := 1;
repeat
 a[ i ] := 3 * position - i;
 i := i + (4 * j);
until i  > (limite - 5);
```

après optimisation :

```
i := 1;
t1 := 3 * position;
t2 := 4 * j;
t3 := limite - 5;
repeat
 a[ i ] := t1 - i;
 i := i + t2;
until i  > t3;
```

Enfin, quelles que soient les optimisations possibles au niveau du code machine, il faut toujours garder à l'esprit que l'amélioration la plus sensible (et la plus intelligente) consiste avant tout à rechercher le meilleur algorithme capable de résoudre un problème particulier. En effet, les différences de performance entre les divers algorithmes sont souvent d'un ordre de grandeur bien supérieur aux gains que l'on peut attendre d'un «bidouillage» du code machine. Mais dans ce domaine, c'est au programmeur de réfléchir et non au compilateur... ■



La gestion des erreurs est un composant essentiel d'un bon compilateur (ici le Turbo C de Borland).

gage objet (ou langage machine) de la machine sur laquelle le programme devra s'exécuter. Le rôle du générateur de code final est en particulier d'assigner des adresses relatives en mémoire à chaque variable et de déterminer les registres du microprocesseur à utiliser pour effectuer les calculs intermédiaires. Ce module est évidemment très dépendant du langage machine final (et de l'ordinateur cible). Par opposition, les autres éléments d'un compilateur sont relativement indépendants de l'architecture cible... du moins en théorie !

Il faut, de plus, noter que la frontière entre l'optimisation du code intermédiaire et la génération du code final est souvent particulièrement floue. En effet, certaines techniques d'optimisation reposent sur un choix judicieux des registres à allouer pour une opération donnée (or l'allocation des registres est normalement du ressort du générateur de code final).

Réciproquement, certaines optimisations locales se font en observant le code final généré au travers d'une «fenêtre» de quelques instructions (peephole optimization). Bien que le code généré à l'étape précédente soit appelé «code final», il lui manque habituellement un certain nombre d'éléments pour être effectivement exécutable par le microprocesseur auquel il est destiné. En effet, l'intérêt de la plupart des langages modernes est de permettre de découper un long programme en plusieurs modules qui sont compilés séparément.

Les adresses

Lors de l'analyse d'un module donné, le compilateur ne peut donc pas fixer de manière définitive les adresses en mémoire des variables et fonctions déclarées dans ce module, puisqu'il ne connaît pas la place nécessaire aux variables et fonctions déclarées dans les autres modules. Pour résoudre ce problème, le code final ne comporte généralement que le déplacement (offset) de chaque identificateur par rapport à l'adresse de base où sera fixé le module courant. C'est ensuite à l'éditeur de liens de fixer les adresses définitives de l'ensemble des modules, ce qui lui permet de déterminer les adresses de tous les identificateurs.

L'éditeur de liens effectue ensuite la résolution des références, opération qui consiste à remplacer dans le «code final» les références aux variables et fonctions par les vraies adresses. Ce n'est qu'à l'issue de cette étape que le programme objet devient réellement exécutable. Pour être tout à fait exhaustif, il reste à décrire brièvement deux composants un peu particuliers en ce qu'ils ne constituent pas une phase spécifique de la compilation, mais interviennent au contraire tout au long de l'analyse du code source par les divers modules. Ces composants sont le gestionnaire de la table des symboles et le gestionnaire d'erreurs.

La table des symboles est utilisée par les autres composants pour constituer la «mémoire» de toutes les définitions rencontrées dans le programme source : les variables et leurs types, les

fonctions et leurs arguments, les tableaux et leurs dimensions, les adresses allouées en mémoire à chaque identificateur, etc. Le gestionnaire de la table des symboles est donc chargé de gérer cette structure afin d'y insérer les informations au fur et à mesure de leur collecte tout en restant capable de les retrouver de manière aussi efficace que possible. La table des symboles est souvent gérée de manière dynamique (on lui alloue de la mémoire en fonction de ses besoins), ce qui complique encore la tâche du gestionnaire. Or, c'est souvent de l'efficacité de la gestion de cette table que dépend la performance globale du compilateur, car tous les modules s'y réfèrent sans cesse.

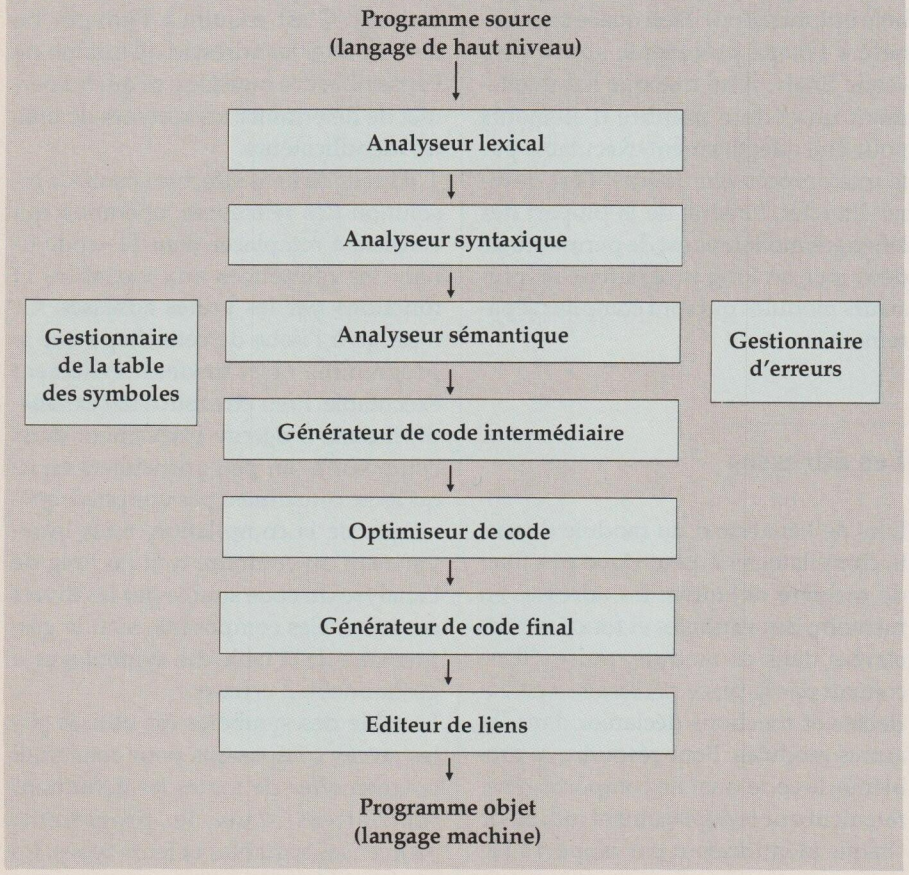
La gestion des erreurs constitue un autre point essentiel pour porter une appréciation sur un compilateur. En effet, un bon compilateur ne doit pas seulement traduire efficacement un code source correct en langage objet, mais également détecter et diagnostiquer aussi précisément que possible les éventuelles erreurs commises par le programmeur. La détection des erreurs dès le stade de la compilation doit être aussi complète que possible, car c'est à ce niveau qu'elles sont le plus visibles et faciles à corriger. Par la suite, une erreur non décelée lors de la compilation pourrait ne resurgir qu'en de très rares occasions... une fois le logiciel commercialisé, par exemple !

Pour des diagnostics de qualité

Quant au diagnostic, son intérêt est d'indiquer au programmeur la cause probable du problème rencontré, afin de l'aider à y remédier.

A cet effet, un bon message d'erreur doit être le plus spécifique et informatif possible. Ainsi, un message précis du type «Il manque une parenthèse fermante» ou «Ce "else" n'est rattaché à aucun "if"» est largement préférable à un simple «Erreur de syntaxe», beaucoup trop général. Le gestionnaire d'erreurs doit également tenter de permettre la poursuite de la compilation afin de pouvoir détecter d'autres erreurs éventuelles, mais sans que l'erreur déjà traitée n'engendre un flot de messages (par exemple : «Ligne 300 : variable "position" non déclarée», «Ligne 301 : variable "position" non déclarée», «Ligne 302 : variable "position" non déclarée», etc.). Ainsi se termine ce tour d'horizon des mécanismes fondamentaux de la compilation. Bien entendu, les détails de l'implémentation de chaque module deviennent rapidement complexes et dépassent de loin le cadre de cet article. Mais il faut savoir que l'écriture complète d'un compilateur pour un «petit» langage n'est pas hors de portée d'un programmeur averti... et l'expérience vaut la peine d'être tentée ! □

LES PHASES SUCCESSIVES D'UNE COMPILATION



L'architecture des compilateurs : bibliographie

- **« Compilers : Principles, Techniques, and Tools »**

Auteurs : Alfred V. AHO, Ravi SETHI, Jeffrey D. ULLMAN
Editeur : Addison-Wesley (1986)

Assez théorique, ce livre couvre cependant tous les aspects de la compilation, depuis l'analyse syntaxique jusqu'à l'optimisation du code généré. Un ouvrage de référence sur le sujet (800 pages).

- **« Compiler Design and Construction »**

Auteur : Arthur B. Pyster
Editeur : Van Nostrand Reinhold (1980)

Ce livre présente, étape par

LA TECHNIQUE DU CACHE-DISQUE : LE TURBO DES DISQUES DURS !

■ Le terme générique "cache" est utilisé en informatique pour désigner un tampon de mémoire qui contient des données identiques à celles stockées à certaines adresses d'un périphérique plus lent. Il existe ainsi des cache-RAM qui interposent de la mémoire vive rapide entre le microprocesseur et la mémoire principale du système, plus lente. La technique du cache-disque qui est étudiée ici fonctionne, quant à elle, en interposant un tampon de mémoire vive (au débit typiquement 10 fois plus important que le disque dur) entre la mémoire principale et le disque dur.

L'intérêt d'un cache est fondé sur le principe de localité, qui affirme qu'un programme exécuté sur un microprocesseur à architecture séquentielle a tendance à accéder de manière répétitive aux mêmes adresses sur un périphérique. Si les données requises sont chargées dans le cache la première fois que le microprocesseur en fait la demande, il y a donc de fortes chances pour qu'il tente à nouveau d'y accéder par la suite. Lors des requêtes suivantes portant sur ces données, le gain de temps sera à peu près égal à la différence entre le temps de réponse du périphérique (lent) et celui du cache (rapide). Bien évidemment, il arrive un moment où le cache est plein.

L'algorithme classique consiste alors à remplacer la zone la plus "ancienne" par de nouvelles données (cet algorithme est souvent désigné par le sigle LRU : Least Recently Used).
Toutefois, certains logiciels tentent de tenir compte de la fréquence d'accès à chaque donnée : une donnée qui est utilisée souvent "vieillit" alors plus lentement que les autres.

Dans le cas particulier du disque dur, de nombreux cas de figure peuvent provoquer des accès répétés aux mêmes secteurs. Ainsi la plupart des appels au DOS nécessitent un accès à la Table d'Allocation des Fichiers (FAT) afin de déterminer l'emplacement du fichier à lire ou à écrire sur le disque. Les secteurs où sont stockées ces informations sont donc d'excellents exemples d'accès répétitif à des données situées à la même adresse. D'ailleurs cette fonction de cache de la FAT est suffisamment importante à elle seule pour que Microsoft l'ait incorporée à partir de la version 4 du MS-DOS dans la commande FASTOPEN. Un autre exemple d'amélioration notable des performances concerne la gestion des bases de données : il arrive en effet souvent que la totalité d'un fichier d'index puisse être chargée dans le cache.

Chaque requête d'un enregistrement ne demande alors qu'un seul accès au disque dur, puisque toutes les recherches sur l'index se font en mémoire vive.
Plus généralement, le temps d'accès à une information située sur le disque dur dépend de 4 facteurs principaux : le temps de déplacement des têtes de lecture/écriture jusqu'à la bonne piste (seek time), le temps de stabilisation des têtes (settling time), l'attente du passage du secteur désiré sous la tête (latency), et enfin le temps de transfert des données de la surface du disque jusqu'à la mémoire vive (transfer time). De tous ces facteurs, le plus critique est le temps de déplacement des têtes de lecture. Les divers algorithmes utilisés pour implémenter un cache-disque se différencient donc par la manière selon laquelle ils cherchent à minimiser le nombre de mouvements des têtes de lecture/écriture du disque dur. La méthode la plus simple (et la plus sûre) consiste à ne faire fonctionner le cache qu'en lecture de données, alors que toutes les écritures sont immédiatement transmises au disque (write-through).

Cependant il est souvent intéressant de conserver plusieurs ordres d'écriture dans le cache, afin d'écrire en une seule fois toutes les modifications qui affectent la même piste du disque dur (delayed writes). Le danger de cette approche réside dans le fait que durant le laps de temps où les ordres d'écriture n'ont pas été exécutés, une panne de courant ou un "plantage" du système sont susceptibles de laisser la structure des fichiers dans un état incohérent...
Pour éviter cela, la plupart des logiciels de cache-disque forcent l'écriture des modifications du cache sur le disque dur à intervalles réguliers (5 secondes, par exemple). Un autre moyen de gagner du temps consiste à lire un certain nombre de secteurs d'avance sur la piste demandée, en anticipant sur un éventuel accès séquentiel au fichier concerné (read-ahead).

La mise en oeuvre d'un cache-disque est donc une technique très versatile (à condition de disposer d'assez de mémoire vive), qui procure souvent un gain de vitesse sensible tout en étant totalement transparente pour l'utilisateur final. De plus, un driver spécifique est fourni avec MS-DOS à partir de la version 4 : SMARTDRV.SYS. Il ne reste alors qu'à prendre goût à la vitesse ! ■

étape, la conception et l'implémentation d'un compilateur pour un sous-ensemble du langage Pascal, avec le code source correspondant. Il propose donc une approche essentiellement pratique du sujet (350 pages).

- **« Compiler Construction for Digital Computers »**

Auteur : David Gries
Editeur : John Wiley & Sons (1971)

Révélant son âge de par sa présentation typographique désuète, cet ouvrage comporte néanmoins des algorithmes précieux et intéressants pour qui doit réellement implémenter un compilateur. L'on remarquera en particulier une bonne étude de la manière de gérer une table des symboles et de programmer un préprocesseur (500 pages). □

PROCESSEURS PARALLÈLES ET DISQUES ARRAYS

ERIC YONTER

A l'origine de ces travaux, il existe une profonde motivation et une soif inlassable de performance. A l'évidence, les solutions proposées n'ont rien de naïf et sont encore moins le résultat de tâtonnements hasardeux. A travers la démocratisation de la recherche industrielle et grâce à une coordination des efforts en vue d'optimiser le fonctionnement du matériel, on aboutit à des formes de systèmes nouveaux. Dans certains cas, l'objectif inavoué de ces architectures nouvelles est de faire ressortir la vraie puissance du matériel potentiel qui n'avait pas été exploité jusqu'à présent. Dans d'autres cas, en revanche, il s'agit simplement d'interconnecter une série de processeurs préconçus pour être facilement reliés. Il est intéressant de constater que ces deux approches sont légèrement différentes bien qu'elles se fixent le même objectif.

Fondamentalement, la technique consiste à mettre ensemble plusieurs centaines de processeurs dans un réseau à couplage dense, d'où l'importance des communications, facteur sur lequel il convient d'insister lourdement. Ensuite il faut associer à ce réseau une batterie de disques plus connue sous le nom de "disk array", digne de cette appellation et hautement performante. Le système n'étant plus accessible par des voies de communication usuelles, il faut également associer un ordinateur dit frontal, qui a pour but de centraliser et les accès et la gestion du système opératoire (système d'exploitation). Souvent, entre le frontal et la machine parallèle, se manifeste un écart important de performances. L'existence du frontal est fortement liée à la notion d'environnement, qui constitue un chapitre à elle seule. On le voit, l'on aborde au fur et à mesure un certain nombre de concepts inhabituels, néanmoins composés d'objets facilement reconnaissables.

Un assemblage complexe de pièces simples

Pour visualiser un de ces systèmes nouveaux, il faut conceptualiser une architecture matérielle différente ou simplement penser autrement lorsqu'il s'agit de projeter l'ordinateur -probablement- de demain. La première condition pour atteindre cet objectif consiste à revoir ou réviser des notions auxquelles nous sommes habitués et qu'il s'agit d'abandonner progressivement

Ainsi allons-nous tenter de nous introduire dans une première boucle épistémologique à deux éléments où se trouvent face à face un matériel à composants délibérément choisis simples d'une part, et une pensée centrée autour de l'idée de traitement massif de l'information d'autre part. En utilisant les pièces du puzzle qui sont le matériel et les idées-moteurs, il faut enrichir cette boucle afin de ramener l'ensemble à un niveau de fonctionnement permettant une utilisation.

Le matériel cité dans cette boucle est évidemment un système informatique au complet. Quant à la pensée, elle est composée d'une suite de réflexions orientée vers l'assemblage des pièces du système final, complexe. Tout au long du parcours, les règles qu'il faut respecter sont la performance, l'accessibilité économique du projet et la facilité d'utilisation du système opérationnel. Dans cette boucle il existe une forte interaction entre les possibilités matérielles et les réflexions dominantes; l'une dirigeant l'autre, ou en corrigeant la trajectoire. Entre ces deux composants, il n'existe pas de prédominance mais une complémentarité.

Les développements techniques récents démontrent que plusieurs axes sont possibles pour la composition des systèmes parallèles. Un des ces axes majeurs est lié à la taille de l'unité centrale -ou taille de grain- qui constitue l'élément insécable dudit système. Le système parallèle final est à son tour composé d'une multitude de ces petits éléments, c'est lui qui forme le deuxième axe principal. A l'opposé de l'indivisibilité de la taille du grain, la taille du système complet est modulable. On peut envisager des systèmes composés de cellules qui traitent de quelques bits, aussi bien que des systèmes comportant des cellules de taille importante. La logique impose, bien entendu que plus la taille du grain est moindre, plus il en faut en grande quantité afin de traiter un problème donné. La proposition inverse se heurte à une barrière économique. Il est en effet difficile d'envisager d'interconnecter des centaines de gros ordinateurs dont le coût unitaire devient important. En reprenant les axes de développement possibles, on peut dire qu'il existe des divergences en ce qui concerne la taille du grain, mais aussi

A l'horizon de l'an 2000 nous percevons progressivement l'émergence des idées et structures nouvelles. Là où on la croyait à bout de souffle et finie, la créativité hardware (le matériel) ressurgit et introduit des solutions quasi révolutionnaires qu'il est nécessaire d'accepter et de mettre à profit. A titre d'exemple, le concept d'architecture paralléle, dans le domaine des systèmes informatiques, est l'objet d'études de plus en plus poussées.

une convergence d'objectifs entre ces différents systèmes lorsqu'il s'agit d'obtenir un certain niveau de performance. En résumé et pour une meilleure compréhension, il s'agit là d'une équation à deux variables : la première variable est liée à la taille du grain, tandis que la seconde est le nombre de cellules constituant le système complet. La puissance globale qui en résulte est fortement dépendante de ces deux facteurs.

La taille du grain mentionnée plus haut ne se résume pas seulement au nombre de bits traités par une cellule. Il faut tenir compte de plusieurs autres facteurs qui peuvent être listés de la façon suivante : une cellule, avant tout, est autonome, et elle est bâtie autour d'une unité de traitement (c'est-à-dire, une unité processeur) et d'une mémoire de taille variable. La cellule doit disposer d'un moyen de communication et c'est là un facteur clé et déterminant du système parallèle, qu'il est nécessaire de détailler. Ensuite on peut mentionner une unité de calcul associée à l'unité centrale (coprocesseur numérique) dont la présence n'est pas obligatoire (par exemple, un système orienté exclusivement vers le traitement des symboles n'a peut-être pas besoin d'une unité de calcul flottant, d'où économie possible).

Ainsi s'achève la liste des éléments statiques et essentiels d'une cellule. Les contraintes principales à ce stade sont, dans un premier temps, l'encombrement physique de la cellule, suivi bien entendu de considérations économiques puisqu'il faut les interconnecter en grand nombre.

Les variations portant sur ces éléments statiques affectent considérablement la taille du grain, et par conséquent la puissance de traitement de la cellule même, qui à son tour affecte la puissance globale du système. Qualitativement et d'une manière générale, il semble légitime de se poser les questions suivantes : faut-il disposer de cellules de petite dimension mais en très grand nombre, ou est-il préférable d'agrandir la taille de la cellule et d'interconnecter un nombre raisonnable d'éléments ? Où se trouve le point d'intersection de ces deux variables ? Quelles seront les conséquences d'un choix précis lors de l'utilisation du système final ? Existe-t-il une combinaison de taille et de nombre de cellules qui soit plus appropriée pour résoudre un problème donné ? Autrement dit, entre la structure d'une machine parallèle et un problème du monde réel, quel est le rapport le plus adéquat et quelles valeurs pertinentes permettent d'obtenir les meilleures performances.

En fait, les éléments sont multiples, et l'ensemble est à la fois complexe et nouveau. Chaque choix détermine durablement les effets de l'objet qui est à construire. Le travail de modélisation et d'abstraction consiste à établir une succession de définitions cohérentes. Dans un premier temps, nous nous concentrerons sur les concepts liés au "disk array". En effet, il est peu probable qu'un tel système puisse se satisfaire d'un mécanisme aléatoire de connexion de disques. L'étude qui suit est composée de deux parties, dont la première aborde les particularités d'un disque qui doit être inséré dans un disque array. La secondepartie, pour sa part, expose quelques architectures de disk array issues des travaux effectués au Berkeley Institut courant 1987 et qui sont toujours d'actualité. Il ne faut pas oublier que les disk arrays sont, en fait, un des multiples composants d'un système parallèle, et que la totalité du sujet est bien plus vaste.

Les spécifications disques pour un "disk array"

L'objet de ce paragraphe est de mettre en évidence les caractéristiques que l'on peut formuler à l'égard d'un disque destiné à être utilisé dans un "disk array" (le terme le plus approprié que l'on puisse proposer étant simplement "systèmes de disques parallèles ou SDP").

La terminologie anglo-saxonne, RAID, est un néologisme composé des initiales des mots "Redundant Array of Inexpensive Drives". Comme l'exprime la formule, il s'agit de disques économiquement accessibles, ce qui permet l'emploi en masse de ces unités. Les principaux critères étant de cibler des périphériques que l'on peut trouver en seconde source (ou plus), avec des bandes passantes élevées, et qui sont susceptibles de supporter un usage
intensif sans que la mécanique accuse une usure accélérée.

Dans le domaine des disques parallèles, on dénombre essentiellement deux grandes techniques. L'une est basée sur la duplication des informations

(mirrored ou shadowed), tandis que l'autre est faite d'une méthode de décalage des enregistrements (striped) à travers les multiples unités. Il existe aussi un mélange subtil des deux (tous ces concepts sont repris et développés en page suivante, dans le chapitre «les diques paralléles»).
Dans le but d'accroître les performances des batteries de disques, une des techniques employées consiste à obtenir la parfaite synchronisation des cycles de rotation des disques impliqués. Cette synchronisation s'obtient en injectant l'impulsion index d'un disque désigné comme maître, dans les circuits électroniques des disques désignés comme esclaves. Au bout d'un certain temps dû à l'inertie mécanique, les disques sont synchronisés en rotation et, de ce fait, leurs impulsions d'index apparaissent pratiquement aux mêmes intervalles de temps. L'avantage de cette technique se voit dans le "temps de réponse" global de la batterie de disques. Autrement dit, les données réparties sur les disques apparaissent au contrôleur extérieur presque simultanément; le taux de transfert des données ainsi parallélisées atteignant des quantités importantes dans une plage de tolérance de temps de la taille d'un demi-secteur.

Insertion/extraction

Une caractéristique très recherchée est la possibilité d'insertion et d'extraction rapide en opérationnel (en fonctionnement), des unités de disques. Le paramètre en question apparaît en fait comme un problème de maintenance. Néanmoins, pouvoir extraire et insérer une unité de disque, dans un ensemble de disques parallèles, via une manipulation simple effectuée par un opérateur, raccourcit le temps d'indisponibilité du système en cas de panne ou autre défaillance.
En second lieu, le fait de pouvoir physiquement soustraire un disque à une base (ou une banque) de données a deux effets de bords qui sont : d'abord la sécurité de l'utilisation, ensuite la possibilité de changer de support et donc de données. Cette dernière suppose que le disque a des renforcements mécaniques au niveau des ses connecteurs et autres parties risquant d'être souvent sollicitées. L'interface électrique a également besoin de quelques aménagements pour prendre en compte les effets de l'extraction et de l'insertion. En contrepartie, le nouveau disque qui vient d'être inséré nécessite une mise à jour rapide de son contenu. Les interfaces les plus performantes sont désormais disponibles dans le domaine des disques courants. Il existe un choix étendu comprenant les systèmes d'interfaçage à double accès pour plus de sécurité en cas de panne, ou de meilleures performances en fonctionnement normal. De plus en plus, les contrôleurs intégrés disposent d'intelligence, pour accroître les facilités d'adressage, et d'algorithmes d'optimisation des recherches de pistes et de secteurs. Ils effectuent également des corrections d'erreurs sophistiquées sur des longueurs importantes de plusieurs bits.
La consommation électrique d'un disque devient un paramètre non négligeable dès lors qu'il s'agit de réunir ensemble plusieurs unités dans un espace réduit. L'accumulation de la chaleur, notamment, est un facteur qu'il faut prendre en compte dès le début d'une étude.
Quant à la stratégie employée pour l'occupation adéquate de l'espace disque, celle-ci génère un phénomène de granularité qui diminue le taux de rentabilité de l'utilisation. Dans un système de disques parallèles, ce facteur est multiplié par le nombre d'unités utilisées. Une stratégie mal adaptée peut donc engendrer une perte considérable de l'espace disque. Un autre phénomène qu'il faut prendre en considération est dû à la capacité sans cesse croissante des disques. Ce phénomène pose le problème d'accès ou de disponibilité d'une très (peut-être trop) grande quantité d'informations sous un seul mécanisme de lecture. Il faut rappeler que l'énorme quantité d'informations -que l'on peut assimiler à un long ruban- passe par un goulot d'étranglement qui est l'interface du disque. La vitesse à laquelle ce passage se déroule est connue comme étant "le taux de transfert des données" et s'exprime en octets par seconde. Au-delà d'un certain seuil de capacité-disque on peut se poser la question de la nécessité d'un rapport si élevé entre le goulot et la longueur du ruban ci-dessus mentionnés.

Performances d'un disque

D'une manière générale, les performances d'un disque comprennent essentiellement les trois facteurs suivants : le taux de transfert des données, le temps d'accès moyen et le "latency time" qui correspond au temps de rotation du disque avant l'arrivée du secteur demandé sous les têtes de lecture/écriture. Selon les applications, certains de ces paramètres peuvent influencer considérablement les performances globales. Par exemple, une application du type OLTP (On Line Transaction Process) est très fortement influencée par le "rotational latency time" (dans le temps d'accès total, la part de ce paramètre est estimée à 70%). Il est donc permis de penser que des disques ayant des vitesses de rotation plus élevées seront mieux adaptés pour ce type d'application.
Par contre, lorsque l'application nécessite des transferts de blocs de taille importante, l'imagerie par exemple, le paramètre dominant devient le taux de transfert des données (en octets par seconde). Dans ce cas précis, la somme des deux autres paramètres n'atteint guère les 10% du temps d'accès total. En contrepartie, l'inconvénient d'une vitesse de rotation élevée est la consommation électrique du disque. Pour une dimension de disque donnée, on estime qu'il existe une certaine limite de puissance admissible (due à la chaleur à dissiper, près de 70 watts pour une unité de 5,25 pouces pleine hauteur). Dans l'état actuel de la technolo-

gie courante, la vitesse de transfert des données à travers l'interface se présente comme étant limitée vers les 40 Mo par seconde (IPI-2 est estimé à environ 100 Mo).
Il est bien évident que cette liste n'est pas exhaustive. Cependant, elle donne un aperçu des principaux paramètres concernant un disque destiné à un système de disques parallèles. Le chapitre suivant décrit les principes des "systèmes de disques parallèles", où les deux axes essentiels et convergents sont le problème des performances et celui de la duplication (redondance) des données.

Les disques parallèles (DISK ARRAYS)

Selon les descriptions en vigueur, un système de disques parallèles (SDP) est composé de deux ou plusieurs disques. Il a pour but d'accroître les performances et d'avoir un maximum de sécurité vis-à-vis des erreurs dues à un mauvais fonctionnement ou à une défaillance technique. Le terme "array" est employé pour désigner n'importe quelle architecture comportant une batterie de disques montés en parallèle et reliés à un système.
Dans tous les cas, il y a parallélisation des unités pour un meilleur débit des données. Par conséquent, hormis la parallélisation qui est commune à l'ensemble, les deux techniques principalement employées sont : la duplication (mirrored ou shadowed) des unités et la technique de décalage des enregistrements (striped) à travers les unités connectées. En fonction de la nature des architectures utilisées, il existe une classification des SDP, qui sont numérotées de RAID I à RAID V.
Au premier niveau, on trouve le RAID I qui est constitué de deux sous-systèmes indépendants. Il s'agit en fait de deux disques et de deux contrôleurs associés à ces unités, l'un étant le miroir de l'autre. Vue de l'extérieur, l'écriture se déroule comme s'il s'agissait d'une seule unité. Tandis que la lecture, en fonction du point de l'observation, peut apparaître comme étant une opération de lecture unique, ou deux lectures provenant de deux unités séparées. Cette dernière possibilité permet au contrôleur extérieur de pouvoir comparer le résultat de ces lectures séparées. A la suite de quoi, soit la lecture est déclarée cohérente, soit l'une des deux unités nécessite la mise à jour de son contenu (ou bien il s'agit d'une panne). Il est bien évident qu'il existe une ligne théorique au-delà de laquelle le système voit une lecture et une écriture uniques.
Dans le deuxième groupe, RAID II, en considérant que l'ensemble est géré par un contrôleur unique, on distingue un groupe de disques chargés de contenir les données et un deuxième groupe supportant les données de correction en cas d'erreur. L'ensemble est connecté en parallèle sur le contrôleur ci-dessus mentionné. Le groupe principal peut être composé de quatre disques, et le groupe secondaire de trois unités du même type (d'autres combinaisons sont possibles). Des interfaces disques larges de huit bits permettent d'obtenir des flux de données de 32 bits associés à un groupe secondaire de 24 bits de correction (check disk hamming, ECC). La bande passante globale est équivalente au produit du nombre d'unités par le taux de transfert d'un disque. Autrement dit, la lecture ou l'écriture doivent correspondre à un groupe d'octets (en fonction de la taille du bloc provenant du disque) multiplié par le nombre de disques.
Le groupe RAID III est une version constituée d'une structure très similaire à la précédente. Il comporte quatre disques de données et un disque pour la parité. Vis-à-vis de l'extérieur, les opérations de lecture et d'écriture apparaissent comme provenant d'une unité unique. La bande passante est également équivalente au produit du nombre d'unités par le taux de transfert d'une unité.
Le groupe suivant, RAID IV, est similaire au groupe précédent du point de vue de la structure physique. La différence entre les deux groupes réside dans l'organisation et la répartition des données sur les unités. A travers le contrôleur, ce quatrième groupe est vu par le système extérieur comme quatre disques indépendants.
Le groupe RAID V est un ensemble d'unités considérées par le contrôleur comme étant des disques individuels avec la duplication répartie des données sur chaque disque. Il existe d'autres arrangements à base de SDP multiples pour des arrays de très large capacité, ainsi que des combinaisons de diverses techniques précitées.

Conclusion

Avec les SDP, des domaines techniques jusqu'à présent réservés aux gros systèmes deviennent abordables. Devenus économiquement accessibles grâce à l'apparition de disques simples et robustes, les systèmes de disques parallèles ouvrent des voies totalement nouvelles. Les performances des ensembles ainsi couplés, on l'imagine, sont de plus en plus remarquables. Néanmoins, les contrôleurs associés à ces disques peuvent dans certains cas manquer d'intelligence. En effet, ils sont fondamentalement orientés vers des échanges massifs de données. Tout en utilisant la technique du parallélisme, il existe d'autres ouvertures technologiques qui mettent à profit une puissance de traitement supplémentaire associée aux contrôleurs disques. Ces techniques pourraient engendrer un impact considérable dans les habitudes de programmation sans pour autant changer de langage informatique. Alors, nouvelles techniques de programmation ou nouveaux langages informatiques ? D'aucun pourraient même penser qu'il s'agit là d'une vision complètement révolutionnaire de l'intelligence informatique... Les ouvertures que l'on peut imaginer, notamment dans des applications, sont immenses. Qu'il s'agisse des bases de données ou de traitement à l'aide des langages à objets, les besoins en performances sont très présents. □

LA GESTION DE LA COULEUR

NORBERT RIMOUX

La différence entre la vision qu'a l'utilisateur de la couleur et les techniques qui président à sa manifestation à l'écran, s'explique par deux raisons. En premier lieu, la couleur d'une image numérique doit être rigoureusement modélisée afin d'être compatible avec un quelconque système de traitement de l'information. D'autre part, chaque élément de la chaîne informatique se caractérise par son propre spectre de sensibilité à la couleur; autant de caractéristiques qui diffèrent parfois grandement de celles de l'oeil. Aussi un opérateur ne pourra-t-il pas juger correctement de la colorimétrie de son image au travers de la simple observation de son moniteur. Dans la pratique, le traitement numérique de la couleur s'apparente parfois à un véritable casse-tête, que seule la conjonction des trois qualités suivantes permet de résoudre :

- Bonne appréciation "artistique" des couleurs,
- Calibration de l'ensemble de la chaîne informatique,
- Bonne connaissance des techniques de l'image couleur.

Le premier de ces critères de réussite, sans doute le plus complexe car subjectif, ne sera pas développé dans le cadre de cet article.

L'œil humain et la couleur

Physiquement, la lumière est un cas particulier de rayonnement électromagnétique, dont la longueur d'onde est comprise entre 4 000 et 7 000 angström (10 -10 m). Cet intervalle correspond d'ailleurs à l'une des deux fenêtres de transparence au rayonnement, de l'atmosphère terrestre, preuve de l'intelligence de l'évolution des espèces. Ces deux limites correspondent respectivement aux deux impressions colorées que nous nommons violet et rouge.

Dans sa logique, le fonctionnement de l'oeil humain n'est pas sans rappeler celui des caméras CCD. En effet, la vision colorée est le fruit de l'interaction entre le rayonnement électromagnétique ambiant et les cellules dites cônes, localisées au centre de la rétine. Les biologistes ont mis en évidence trois types de cônes, chacun étant sensible à un domaine lumineux : bleu, vert et rouge. En d'autres termes, chaque type de cône n'est sensible qu'à un intervalle du spectre visible.

A l'instar de tous les détecteurs, la sensibilité d'un cône n'est pas constante sur l'ensemble de son intervalle utile. Il découle de cette remarque que, à intensité de rayonnement égale, certaines longeurs d'onde seront perçues comme plus ou moins lumineuses. Ce phénomène, valable pour chaque type de cône, est cumulatif, d'où une sensibilité de l'oeil très variable en fonction de la longeur d'onde de la lumière (*cf figure 1*). Si l'on prend le cas d'une lumière monochromatique, c'est-à-dire qui ne rayonne que dans une seule longueur d'onde, on s'aperçoit que la sensibilité de l'oeil atteint son maximum aux alentours de 5 500 angström, ce qui correspond approximativement à un jaune, et baisse très vite aux environs des extrémités du spectre visible (bleu et rouge). D'autre part, les longueurs d'ondes les plus courtes (vers le bleu) sont celles qui paraissent les moins lumineuses.

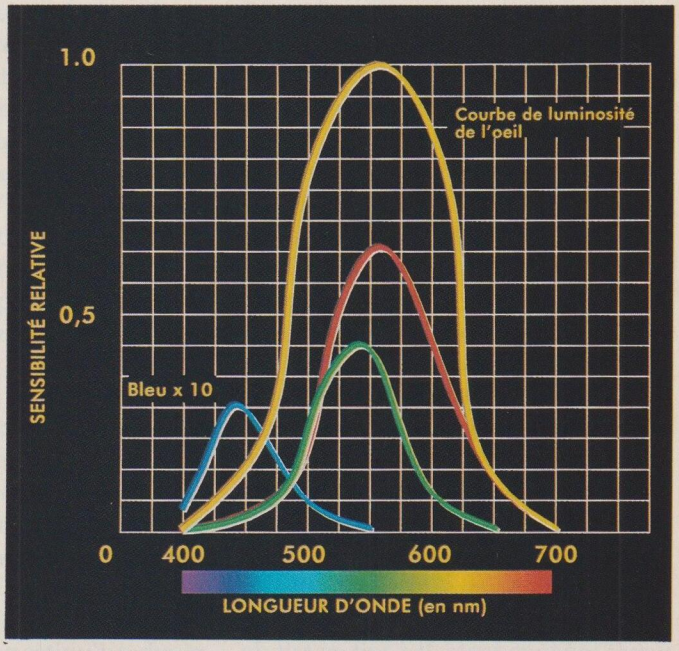


Figure 1 : sensibilité de l'oeil en vision diurne

A la lueur de ces considérations, une conclusion rapide énoncerait qu'une couleur donnée résulte de la superposition de ses composantes verte, rouge et bleue. Une telle conclusion,

Notion naturelle s'il en est, la couleur nous est si familière qu'il ne paraît pas nécessaire de l'expliquer. A l'instar de l'odorat ou du toucher, la première démarche qui vient à l'esprit dès lors qu'il s'agit de composer avec elle, consiste à entrainer ses sens afin de percevoir au mieux les diverses nuances qui la composent, ainsi que la qualité visuelle de leurs interactions. Toutefois cette approche montre bien vite ses faiblesses lorsqu'il s'agit de traiter une image numérique.

sans être fausse, serait pour le moins hâtive et surtout peu complète. D'une part, les différents cônes ne sont pas sensibles à une seule longueur d'onde mais à un intervalle, et d'autre part, l'oeil ne constitue qu'un récepteur physique. La sensation de la couleur est fabriquée par le cerveau, à partir des mesures qu'il reçoit de l'oeil. Aussi, la méthode d'interprétation qu'il met en oeuvre contribue-t-elle grandement aux caractéristiques de la vision humaine.

Comme en témoigne le langage courant, la perception naturelle d'une couleur chez l'homme ne fait pas appel à une logique de mélange. La description des couleurs est basée sur les notions de teinte, de pureté de la couleur, et de luminosité. Telle couleur sera décrite par un observateur comme "un bleu cyané, pastel, très clair". On retrouve dans cette sentence, une quantification des trois caractéristiques que nous venons d'introduire : la dominante bleue, une certaine impureté qualifiée de pastel, et un niveau d'intensité très élevé. D'une manière plus générale, nous pouvons caractériser une couleur par sa teinte, c'est-à-dire le qualificatif donné à cette couleur (rouge, bleu, vert, jaune, etc..), sa saturation, c'est-à-dire son degré de dissolution dans le blanc, et sa luminosité. Une couleur pure sera saturée à 100%. Si la saturation atteint 0%, il n'existe plus de dominante colorée, et la lumière perçue est dite achromatique (absence de couleur), ce qui se traduit par une nuance de gris, entre le noir et le blanc, en fonction de sa luminosité. Le cerveau agit donc comme un intégrateur qui perçoit trois quantités physiques qui sont : la longueur d'onde dominante (teinte), sa contribution à la luminosité de l'ensemble (saturation), et la luminosité d'ensemble de la lumière perçue (intensité par unité de surface).

Point de vue expérimental

Une expérience pratique simple consiste à éclairer un support quelconque avec une lumière donnée. Un observateur doit alors tenter de reproduire la même valeur, sous la forme d'une tache colorée adjacente, à l'aide d'une ou plusieurs sources monochromatiques (une seule longueur d'onde) dont il peut faire varier l'intensité. De nombreuses expériences de ce type ont mis en valeur plusieurs propriétés intéressantes du système visuel humain. Si l'observateur ne dispose que d'une seule source monochromatique, il ne pourra apparier sa lumière test avec la référence que si cette dernière est de la même longueur d'onde. Toutefois, il pourra toujours obtenir l'intensité lumineuse équivalente. En revanche, seule l'utilisation de trois sources différentes lui permettra d'apparier la quasi-totalité des lumières testées, à condition que les lumières étalons soient suffisamment éloignées l'une de l'autre dans le spectre visible. Ce schéma correspond donc respectivement à des sources de longueur d'onde faible (bleu), moyenne (vert) et élevée (rouge). Ce cas général est pourtant battu en brèche pour certaines couleurs. Le cas d'un bleu-vert, notamment, met en évidence le fait que l'observateur parvient à obtenir la bonne teinte, mais elle lui paraîtra trop lumineuse. La seule solution consiste alors à ajouter du rouge dans la lumière de test afin de la rendre plus lumineuse (*figure* 2). Ces expériences montrent que la quasi-totalité des couleurs peuvent être créées à partir de trois sources monochromatiques différentes, R (rouge), G (vert),

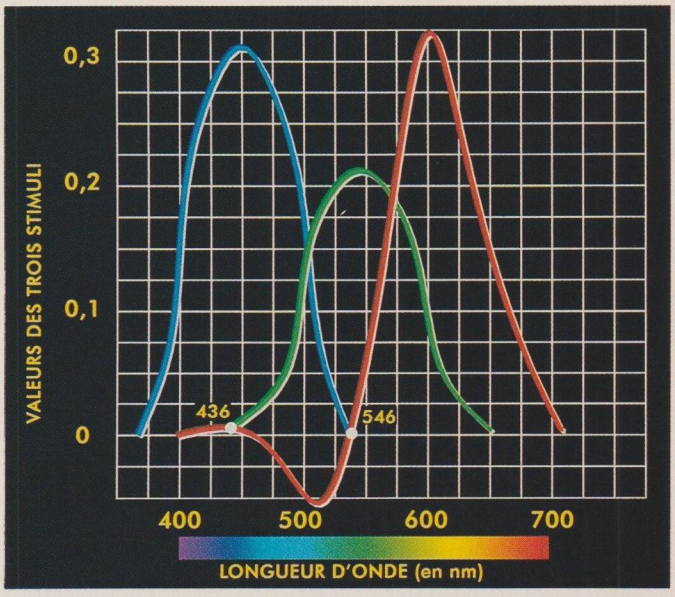


Figure 2 : Courbe se synthèse additive expérimentale

B (bleu). Pour un jeu de sources étalons R-G-B donné, une couleur C sera décrite mathématiquement par la relation :

C = rR + gG + bB, où r,g,b sont des proportions (de 0 à 1).

Le cas du cyan (bleu-vert) correspond à un coefficient "r" négatif, ce qui n'a bien sûr aucune signification physique, la "lumière négative" restant à inventer. D'autre part, les scientifiques on constaté que les proportions de r,g,b sont constantes pour un jeu de sources étalons particulier, quelle que soit l'intensité de la lumière à simuler. Cela revient à dire que si l'on double l'intensité des sources, l'intensité de la lumière de référence perçue double elle aussi, soit la relation :

2C = 2rR + 2gG + 2bB.

Généralement, une même couleur de référence peut être obtenue à l'aide de trois sources différentes d'une première expérience, avec bien entendu de nouveaux coefficients r,g,b. Les couleurs d'étalonnage sont alors appelées métamères l'une de l'autre. En pratique, cela revient à dire qu'une couleur quelconque pourra être obtenue par la combinaison de deux sources composées différentes.
Trois types de cônes dans l'oeil, trois sensations visuelles attachées à chaque couleur, trois sources lumineuses monochromatiques pour synthétiser expérimentalement une couleur : autant de constats qui permettent de déduire la nature tridimensionnelle de la couleur. Toutes ces caractéristiques sont contenues dans les lois de Grassman.

- Le système visuel distingue trois stimuli différents.

- Quatre couleurs sont toujours reliées linéairement. Il en découle que si C=C1 et C=C2, alors C1=C2.
D'autre part,
C1 + C2 = r1R1 + r2R2 + g1G1 + g2G2 + b1B1 + b2B2.
Dans le cas particulier où les sources sont identiques,
C1 + C2 = (r1+r2)R + (g1+g2)G + (b1+b2)B.

- L'espace des couleurs en trois dimensions est continu. La variation continue d'une composante entraînera une variation continue de la couleur résultante.

Une vision pragmatique de la couleur

En moyenne, le système visuel humain s'avère capable de discerner quelque 350 000 couleurs différentes. Comme on peut maintenant s'en douter, elles ne sont pas toutes égales entre elles, en raison de la sensibilité non uniforme de l'oeil. Ainsi, à saturation et luminosité constante, le système visuel humain est capable de discriminer des teintes correspondant à une variation de longueur d'onde de l'ordre de 10 angström dans la partie jaune du spectre, intervalle de sensibilité maximal (*cf figure 1*). Ce pouvoir de résolution chromatique se dégrade assez rapidement pour avoisiner les 100 angström au voisinage des extrémités du spectre visible. Au total, le nombre des teintes discernables est de l'ordre de 128. Côté saturation, les performances sont plus médiocres, puisqu'un individu moyen détecte environ 20 saturations différentes par teinte, avec une sensibilité accrue dans les teintes rouges-violacées. La perception des couleurs de l'environnement se fait au travers de la lumière directement émise par une source quelconque, mais aussi par le biais de la lumière réfléchie par les objets. Quelques règles simples existent, qui régissent la couleur apparente d'un objet. Si celui-ci réfléchit plus de 80% de la lumière incidente uniformément, il paraîtra blanc sous un éclairage blanc, ou prendra la teinte de la source illuminante. La teinte d'un objet observé par le biais d'un illuminant blanc est due à l'absorption sélective de certaines longueurs d'ondes. Dans de telles conditions, il apparaitra comme vert si sa surface ne réfléchit pas les rayonnements bleus et rouges. Conséquence de cette remarque, ce même objet éclairé par une source rouge, apparaitra noir. Les caractéristiques complètes de réflection diffuse, spéculaire et transmission de la lumière jouent d'ailleurs un rôle capital en matière de synthèse d'image, et ont donné lieu à de nombreux modèles de rendu, tels que Gouraud ou Phong. La sensation colorée dépend donc étroitement des conditions d'observation de l'environnement, et notamment des caractéristiques chromatiques de l'illuminant. L'influence de la lumière ambiante est moins nette dans le cas d'une source lumineuse, tel un moniteur couleur, à condition que sa contribution à l'énergie lumineuse perçue par l'oeil reste plus faible que celle de l'objet émetteur.

Un modèle universel

En 1931, la Commission Internationale de l'Eclairage (CIE) a établi un modèle universel de représentation de la couleur. La notion de trois composantes suggère immédiatement la représentation d'une couleur par le biais d'un repère orthonormé 3D. C'est cette représentation de base qui a été retenue par le CIE. D'autre part, cet organisme a défini trois couleurs primaires X, Y et Z, de manière à éliminer le problème des valeurs négatives de la figure 2. Il s'agit là de couleurs primaires théoriques. Le choix du CIE permet d'obtenir un triangle XYZ contenant l'ensemble du spectre visible (*figure 3*). D'après ce modèle, une quelconque couleur s'exprimera comme un vecteur de coordonnées X (rouge), Y (vert) et Z (bleu), correspondant à chaque primaire du CIE. Afin de rendre universel son modèle, le CIE a préconisé l'usage de coordonnées normalisées (entre 0 et 1) qui ne sont autres que les proportions de chaque composante primaire nécessaires à l'obtention de la teinte désirée. Ces coordonnées normalisées sont respectivement notées x, y, et z. Les équations régissant ces valeurs en fonction des coordonnées X, Y et Z sont

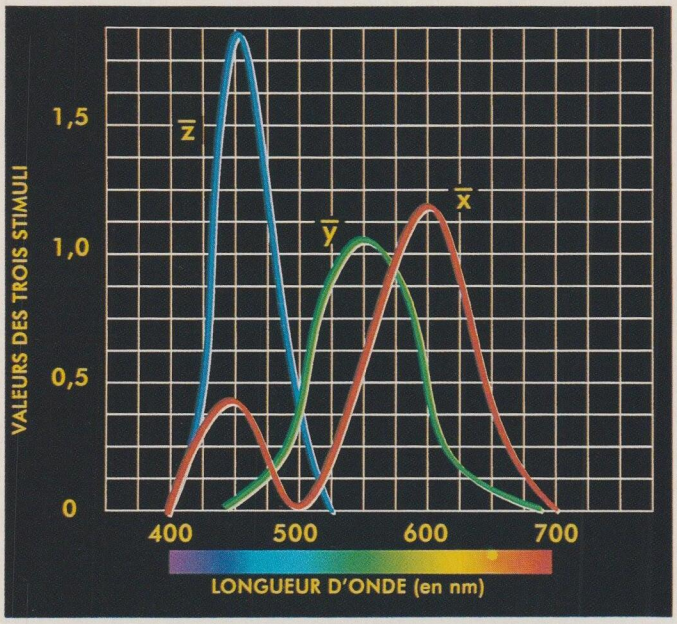


Figure 3 : Sources CIE normalisées

évidemment :

$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+Z}$$

avec $x + y + z = 1$

x, y et z sont appelées les coordonnées de chromacité. Il découle des équations précédentes que la seule donnée de x et y suffit à déterminer la teinte voulue, puisque

$z = 1 - x - y.$

Seule manque la valeur de l'intensité de la couleur (luminance). Par convention, elle est incluse dans la coordonnée Y (vert).

Ainsi, une couleur sera toujours définie par ses coordonnées x,y,Y du CIE.

On obtient la transformation des coordonnées de chromacité CIE en couleurs primaires CIE par les relations suivantes :

$$X = x\,\frac{Y}{y} \qquad Y = Y \qquad Z = (1-x-y)\,\frac{Y}{y}$$

Dernier point important, le triangle XYZ est conçu afin que des valeurs égales des trois couleurs primaires produisent du blanc.
La figure 5 (*cf. page suivante*) représente l'ensemble des couleurs visibles du spectre au sein d'un diagramme de chromacité CIE.
Il est clair que le système CIE n'est pas très bien adapté à un usage interactif. Comment traduire un bleu turquoise pastel en coordonnées CIE ? Toutefois, la normalisation apportée par le CIE est précieuse puisqu'elle offre une base solide pour la conversion des couleurs d'une représentation à une autre, et surtout, elle s'avère être un système de représentation de la couleur totalement indépendant des conditions d'éclairement, ou des périphériques, pourvu que l'on connaisse les caractéristiques CIE de ces contraintes. Outre le modèle lui-même, le CIE a défini les caractéristiques d'un certain nombre de sources (illuminants) standards :

A (0,448 , 0,408) couleur à 2856 K d'un filament de tungstène.
B (0,349 , 0,352) soleil au méridien.
C (0,310 , 0,316) ciel couvert à midi.
D6500 (0,313, 0,329) rayonnement du corps noir à 6504 K.

Notons que la source D6500 constitue le blanc de référence le plus courant en matière de périphérique de visualisation. Plus on s'éloigne du blanc de référence, plus la couleur est saturée (pure).

Les modèles de couleurs numériques

L'informatique en général n'utilise pas réellement le système de représentation des couleurs CIE, ou plus exactement, l'utilisateur n'y a que très peu souvent accès. En fait, la codification d'une couleur en infographie met en oeuvre trois représentations générales :

système RGB
système CMY et son dérivé CMYK
système HSL et son dérivé HSV

Le premier de ces trois systèmes est aussi le plus universellement implanté, car il correspond exactement au fonctionnement d'un moniteur couleur ou d'une carte graphique. Dans ce cadre précis, une couleur est définie par une valeur attribuée à chacune des composantes fondamentales. Généralement, celle-ci oscille entre 0 et 255, soit un octet. Il est ainsi possible de modéliser quelque 16,7 millions de couleurs différentes, valeur amplement suffisante eu égard aux performances du système visuel humain. A la lueur de nos remarques précédentes, il est bien évident que le système RGB correspond à une synthèse additive de la couleur : si les trois composantes sont maximales, la couleur résultante est un blanc. Si les trois composantes sont égales, on obtient un gris plus ou moins foncé. L'accroissement continu d'une seule composante entraîne un accroissement proportionnel de la luminosité d'ensemble. Quoiqu'il ne soit pas vraiment naturel, il se montre assez facile à dominer. Seul problème, le sys-

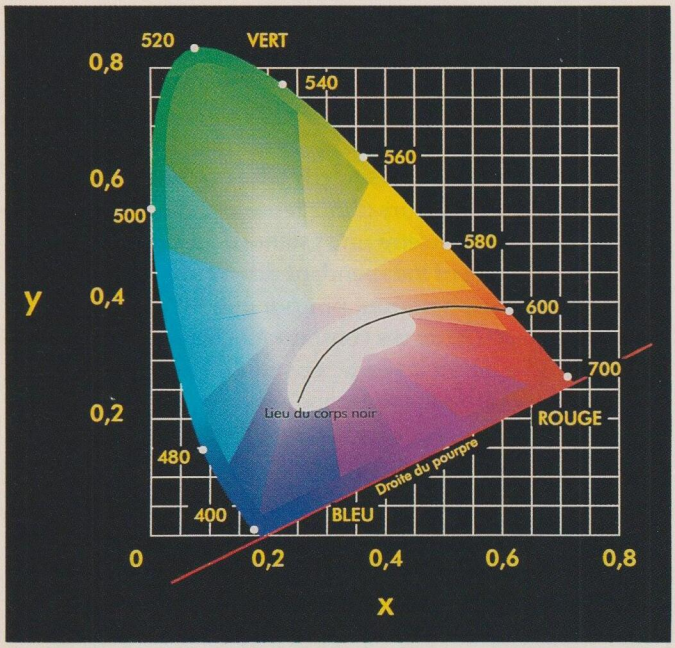


Figure 5 : Diagramme de chromacité CIE (x,y) représentant l'ensemble des longueurs d'ondes visibles.

tème RGB tel qu'il est utilisé en informatique n'est absolument pas indépendant des périphériques, ce qui provoque des dérives de couleur parfois drastiques entre la visualisation écran, et les diverses éditions. Un élément clé de la correction des dérives de couleur réside dans la conversion de coordonnées RGB en leurs équivalents CIE. Ce processus est encore rarement pris en compte par les logiciels d'arts graphiques les plus répandus. Toutefois, le besoin de fidélité devient à ce point impérieux que tous les protagonistes du marché cherchent à résoudre cette épineuse difficulté. L'une des premières tentatives standards est à porter au crédit des concepteurs du format de fichier TIFF (Tagged Image File Format). En effet, la version 5.0 de ce format d'images en mode point inclut à la fois la définition de l'illuminant de référence du fichier, et également les courbes de correction colorimétriques associées. TIFF se base d'ailleurs complètement sur le modèle du CIE.

La phase de conversion CIE vers RGB exige généralement quatre caractéristiques du périphérique cible :

Les caractéristiques x et y CIE des couleurs fondamentales RGB du périphérique, ainsi que les caractéristiques x,y,Y du blanc de référence. Une implantation en langage C des routines de conversion pourrait prendre la forme suivante :

```
/* variables globales */
double xr,yr;              /* primaire périphérique rouge */
double xg,yg;              /* primaire périphérique verte */
double xb,yb;              /* primaire périphérique bleu  */
double xblanc,yblanc,Yblanc;  /* blanc de référence */
```

```
/* pour une visu couleur, les valeurs usuelles sont */
/*                                                  */
/* xr = 0.628   yr = 0.346                          */
/* xg = 0.268   yg = 0.588                          */
/* xb = 0.150   yb = 0.070                          */
/* le blanc de référence étant la source D6500 CIE  */
/* soient :                                         */
/* xblanc = 0.313  yblanc = 0.329  Yblanc = 1.0     */



CIE_to_RGB (double x,double y,double Y,int *R,int *G, int
*B)
 {
   double D,Cr,Cg,Cb,X,Z;
   double a,b,c;        /*coef de la matrice de transforma-
tion*/

   /* Cr égal à Xr + Yr + Zr, coordonnées de la primaire */
   /* rouge du périphérique. même remarque pour Cg et Cb
*/


   D  = xr*(yg-yb) + xg*(yb-yr) + xb*(yr-yg);

   Cr = xblanc*(yg-yb) - yblanc*(xg-xb) + xg*yb-xb*yg;
   Cr*=(Yblanc/yblanc); /* DCr*/

   Cg = xblanc*(yb-yr) - yblanc*(xb-xr) - xr*yb+xb*yr;
   Cg*=(Yblanc/yblanc); /* DCr*/

   Cb = xblanc*(yr-yg) - yblanc*(xr-xg) + xr*yg-xg*yr;
   Cb*=(Yblanc/yblanc); /* DCr*/

   /*calcul des deux coordonnées XZ manquantes */
   X = x*Y/y;
   Z = (1-x-y)*Y/y;


   /*calcul de la composante rouge */
   a = ( (yg-yb) - xb*yg + yb*xg ) / Cr;
   b = ( (xb-xg) - xb*yg + yb*xg ) / Cr;
   c = ( xg*yb - xb*yg) / Cr;

   *R = (int) (a*X + b*Y + c*Z);

   /*calcul de la composante verte */
   a = ( (yb-yr) - xr*yb + yr*xb ) / Cg;
   b = ( (xr-xb) - xr*yb + yr*xb ) / Cg;
   c = ( xb*yr - xr*yb) / Cg;

   *G = (int) (a*X + b*Y + c*Z);
```

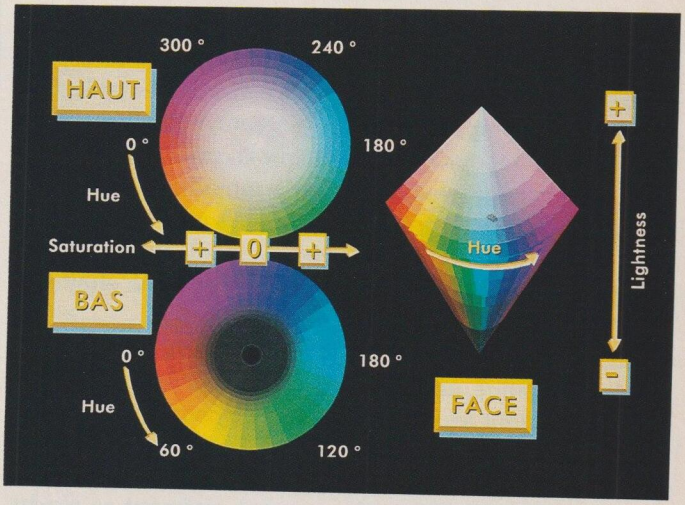
```
    /*calcul de la composante verte */
a = ( (yr-yg) - xg*yr + yg*xr ) / Cb;
    b = ( (xg-xr) - xg*yr + yg*xr ) / Cb;
    c = ( xr*yg - xg*yr) / Cb;

    *B = (int) (a*X + b*Y + c*Z);

  }  /* fin CIE_to_RGB */



RGB_to_CIE (int R ,int G,int B, double *x, double *y, double *Y)
{
    double D,Cr,Cg,Cb,X,Z;
    double a,b,c;          /*coef de la matrice de transformation*/

    /* Cr égal à Xr + Yr + Zr, coordonnées de la primaire */
    /* rouge du périphérique. même remarque pour Cg et Cb */

    D  = xr*(yg-yb) + xg*(yb-yr) + xb*(yr-yg);
    D  = Yblanc/(D*yblanc);

    Cr = xblanc*(yg-yb) - yblanc*(xg-xb) + xg*yb-xb*yg;
    Cr*=D; /* DCr*/

    Cg = xblanc*(yb-yr) - yblanc*(xb-xr) - xr*yb+xb*yr;
    Cg*=D; /* DCg*/

    Cb = xblanc*(yr-yg) - yblanc*(xr-xg) + xr*yg-xg*yr;
    Cb*=D; /* DCb*/

    /*calcul de la composante X*/
    a = xr*Cr;
    b = xg*Cg;
    c = xb*Cb;
    X = a*R + b*G + c*B ;

    /*calcul de la composante Y*/
    a = yr*Cr;
    b = yg*Cg;
    c = yb*Cb;
    *Y = a*R + b*G + c*B ;

    /*calcul de la composante Y*/
    a = (1-xr-yr)*Cr;
    b = (1-xg-yg)*Cg;
    c = (1-xb-yb)*Cb;
    Z = a*R + b*G + c*B ;

    D = X + *Y + Z;

    /*calcul de x et y */
    *x = X/D;
    *y = *Y/D;

  } /* fin de RGB_to_CIE */
```

L'algorithme employé n'est en fait qu'un simple calcul de matrice de changement de repère, et de son inverse, du type :

$$\begin{vmatrix} X \\ Y \\ Z \end{vmatrix} = \begin{vmatrix} Xr & Xg & Xb \\ Yr & Yg & Yb \\ Zr & Zg & Zb \end{vmatrix} \begin{vmatrix} R \\ G \\ B \end{vmatrix}$$

Où Xr, Yr et Zr représentent les coordonnées CIE de la primaire rouge du périphérique (respectivement verte et bleue).

L'utilisation conjointe CIE et RGB permet par exemple de calculer précisément le rendu couleur entre deux périphériques différents. Il suffit de convertir chaque couleur en son équivalent CIE en tenant compte des caractéristiques du périphérique source, puis de procéder à l'opération inverse avec utilisation des propriétés du périphérique de destination. Toutefois, on s'aperçoit que l'emploi des deux fonctions C précédentes n'est guère efficace, en termes de temps de calcul, puisqu'elles font intensivement appel à des variables en point flottant. L'astuce la plus évidente consiste à se fabriquer une table de 256 couleurs, soit 768 valeurs à calculer, qui servira de LUT (Look Up Table) de correction, bien plus rapide à appliquer à une image. Ce principe est d'ailleurs à



Synthèses additives et soustractives.
Représentation par leurs cubes de couleurs respectifs.

la base des fonctions sophistiquées de gestion de la couleur dont dispose le format de fichier TIFF 5.0. Les calculs d'ajustement portent toujours sur la LUT de correction, et non sur l'intégralité de l'image, sauf en cas d'édition ou de visualisation globale de celle-ci.

CMYK, les plus de la synthèse soustractive

Fondamentalement, il n'existe guère de différence entre les systèmes CMY (Cyan, Magenta, Yellow) et RGB. Tous deux sont établis à partir de trois composantes fondamentales. Il en résulte que les calculs généraux réalisés en mode RGB restent valables dans ce nouvel espace. Qui plus est, on peut dire que CMY est directement dérivé de RGB. Sa raison d'être vient du fait que l'on perçoit la couleur soit à partir d'une source lumineuse, auquel cas la couleur perçue résulte de l'addition de trois composantes primaires (systèmes RGB), soit de la réflexion ou de la transmission par un objet quelconque, qui n'est pas lui-même lumineux. Il se produit alors un phénomène d'absorption de certaines fréquences, dont une élimination de certaines teintes. Il devient alors commode de coder la couleur en fonction de cette atténuation, et l'on peut dire qu'une couleur est le fruit d'une soustraction sélective, ou synthèse soustractive.
Les trois fondamentales utilisées sont le cyan, le magenta et le jaune, les complémentaires du système RGB. La relation existant entre les deux modèles s'en déduit donc très simplement :

```
+   +     +   +   +   +
| R |     | 1 |   | C |
| G |  =  | 1 | - | M |
| B |     | 1 |   | Y |
+   +     +   +   +   +
```

Toutes les quantifications de l'espace RGB sont donc entièrement inversées dans l'espace CMY. L'origine de cet espace est le blanc, et le lieu des composantes maximales est le noir. Les procédés d'impression sur papier font naturellement appel à la synthèse soustractive, puisque la couleur perçue provient de la transmission, et, dans une moindre mesure, de la réflexion d'une lumière incidente à travers une encre colorée. Celle-ci filtrera donc la lumière blanche incidente. En théorie, la superposition d'encre cyan, magenta et jaune éliminera toute la lumière, et produira donc le noir.
L'extension classique CMYK (K pour noir), ou synthèse quadrichromique, est rendue nécessaire par des contraintes techniques, dont les caractéristiques physiques imparfaites des encres. Dans la mesure où le noir, et plus généralement les niveaux de gris, sont obtenus par superposition des trois primaires en quantité égale, on peut aisément se rendre compte qu'une quelconque couleur CMY peut toujours se décomposer de la manière suivante :

```
+   +     +     +   +   +
| C |     | C-K |   | K |
| M |  =  | M-K | - | K | avec C >= K, M >= K, Y >=K
| Y |     | Y-K |   | K |
+   +     +     +   +   +
```

De ces quatre relations il découle que la composante noire maximale correspond à "minimum (C,M,Y)". D'où :

```
+   +     +                 +   +    +
| C |     | C - min(C,M,Y) |   | C' |
| M | ->  | M - min(C,M,Y) | = | M' |
| Y |     | Y - min(C,M,Y) |   | Y' |
+   +     | min(C,M,Y)     |   | K  |
          +                 +   +    +
```

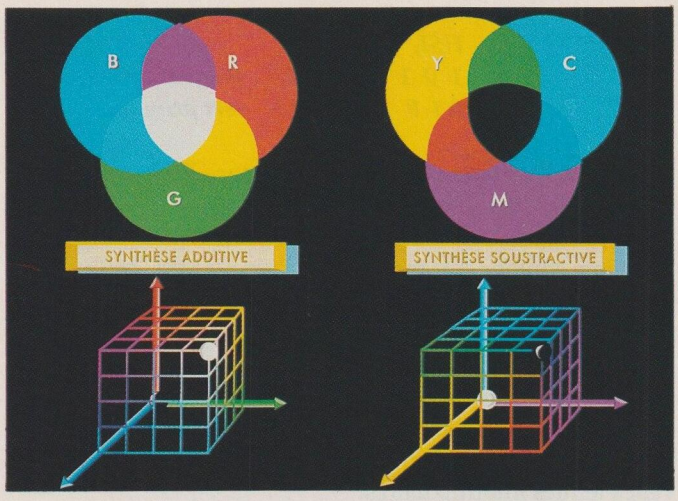
Par exemple, la couleur C (0.5, 0.2, 0.8) CMY se décomposera en C'(0.3, 0, 0.6, 0.2) dans le système CMYK.

HSL, un modèle plus "naturel"

Les deux modèles étudiés jusqu'ici sont basés sur l'utilisation de trois couleurs primaires. Ce schéma correspond bien au fonctionnement de l'oeil. Mais, ainsi que nous l'avons déjà évoqué, les trois stimuli pris en compte par le cerveau sont plus complexes, puisqu'il s'agit de la teinte, de la saturation et de la luminosité d'une couleur. Un modèle informatique, HSL (Hue Saturation Lightness) a été développé dans le but de faciliter la sélection d'une couleur grâce à une logique à laquelle nous sommes tous accoutumés. L'espace des couleurs HSL est représenté par un double cône dont la base est commune. La position angulaire d'une couleur correspond à sa teinte, avec pour valeurs repères celles des 6 principales teintes fondamentales :

Angle	*teinte*
0	*rouge*
60	*jaune*
120	*vert*
180	*cyan*
240	*bleu*
300	*magenta*
360 (0)	*rouge*

La distance par rapport à l'axe de rotation vertical du cône indique, quant à elle, le degré de pureté de la couleur (saturation) : maximale vers la périphérie, nulle au centre. On en déduit que le cas de l'achromatisme (niveau de gris) est représenté par l'axe de rotation vertical des deux cônes. Enfin, la coordonnée verticale permet de quantifier la luminosité



Schématique du modèle HSL

de la couleur : sombre vers le bas, clair vers le haut.
Bien entendu, les couleurs pures se situent toutes dans le plan de base des deux cônes, et sont localisées sur la circonférence de ce cercle.
L'un des grands atouts du système HSL réside dans la facilité avec laquelle un opérateur peut intuitivement modifier une caractéristique globale de l'image, la luminosité notamment. En outre, l'utilisation de HSL dans le cadre d'un logiciel du type "paint" par exemple, facilite l'implantation de modes de tracés particulièrement utiles, dont la recolorisation d'une image monochrome. En pratique, l'utilisation conjointe d'un système à couleurs primaires (RGB ou CMY) et du modèle HSL offre une excellente approche de la gestion informatique des couleurs. La conversion entre les deux modèles peut être considérée comme une tranformation géométrique entre un cube RGB et le double cône HSL. Corollaire de cette conversion, il est aisé de transformer des coordonnées HSL en coordonnées xyY normalisées du CIE. Une implantation C de ces routines pourrait prendre la forme suivante :

```
/* ENTREE : hue entre 0 et 359, sat light entre 0 et 255 */
/* SORTIE : rouge vert bleu entre 0 et 255 */

HSL_to_RGB(int hue,int sat,int light,int *bleu,int *vert,int
*rouge)
{
 double R,G,B,H,L,S,Val,M1,M2;

 /* Normalisation de S et L (valeur entre 0 et 1 */
 L=light/255.0;
 H=(double)(hue);
 S=sat/255.0;

 if (L0.5)       /* cône infèrieur (sombre) */
   M1=L*(1-S);
 else
   M1=L+S-L*S;     /* cône supèrieur */
 M2=2*L-M1;

 /* Cas de l'achromatisme, niveaux de gris */
 if (S==0)
 {
  *rouge=*vert=*bleu=(int)(L*255.0);
  return(0);
 }

 Teinte(H+120,M1,M2,&Val);
 *rouge=(int)(Val*255);

 Teinte(H,M1,M2,&Val);
 *vert =(int)(Val*255);

 Teinte(H-120,M1,M2,&Val);
 *bleu =(int)(Val*255);
} /* fin HSL_to_RGB */


/* Isolation de la composante prépondérante, en fonction
de HUE */

Teinte(double H,double M1,double M2,double *Val)
{
 if (H)  H+=360.0;
 if (H360) H-=360.0;
 if (H<)          *Val=M1+(M2-M1)*H/60;
 if (H=60  && Hœ)  *Val=M2;
 if (H=180 && H)  *Val=M1+(M2-M1)*(240-H)/60;
 if (H=240 && H360)  *Val=M1;
}/* fin Teinte */



/* ENTREE : rouge vert bleu entre 0 et 255 */
/* SORTIE : sat light entre 0 et 255, hue entre 0 et 359 */

RGB_to_HSL(int bleu,int vert,int rouge,int *hue,int
*sat,int *light)
{
 double H,S,L,R,G,B,M1,M2,CR,CG,CB;

 /* Normalisation des coordonnées rouge, verte et bleue (0
à 1)
*/
 R=rouge/255.0;
 G=vert/255.0;
 B=bleu/255.0;

 H=L=S=0;

 /* Evaluation des deux valeurs extrèmes des primaires
RGB */
```

```
/* M1 maximum, M2 minimum */
M1=M2=R;
if(M1G) M1=G;
if(M1) M1=B;
if(M2G) M2=G;
if(M2B) M2=B;

/* Calcul de la luminosité (ce n'est pas la luminance) */
L=(M1+M2)/2;

/* Calcul de la saturation */
/* Cas de l'achromatisme, pas de composante dominante,
donc gris */
if (M1==M2)
 {
  *sat=0;          /* Saturation nulle */
  *light=(int)(L*255);
  return(0);
 }
else
{
  if (L0.5)        /* Cône supèrieur ou infèrieur */
          S=(M1-M2)/(M1+M2);
  else
          S=(M1-M2)/(2-M1+M2);
}

/* Calcul de la teinte */
CR=(M1-R)/(M1-M2);
CG=(M1-G)/(M1-M2);
CB=(M1-B)/(M1-M2);
if (R==M1) H=CB-CG;
if (G==M1) H=2+CR-CB;
if (B==M1) H=4+CG-CR;

H=H*60;
if (H) H+=360.;      /* Angle compris entre 0 et 360 */
*hue=(int)(H);
*sat=(int)(S*255.);
*light=(int)(L*255.);

}/* fin RGB_to_HSL */
```

Pour peu que la précision des calculs soit correcte, il est toujours possible de convertir une couleur entre le modèle RGB et HSL. Mais la transformation entre les deux espaces n'est pas une transformation affine, comme dans le cas RGB à CMY par exemple. Cette remarque mathématique implique que les calculs d'extrapolation entre deux couleurs ne produiront pas le même résultat selon que les calculs seront effectués dans l'un ou l'autre de ces espaces. Les utilisateurs de logiciels de dessin sont d'ailleurs bien au fait de cette contingence. L'exemple suivant illustre ce point.
Soit deux couleurs extrêmes d'un dégradé linéaire :

RGB	*HSL*	
C1	*1 0 1*	*300 1 0,5* *magenta primaire*
C2	*0 1 0*	*120 1 0,5* *vert primaire*

La couleur médiane calculée dans l'espace RGB sera :
C3 = (0.5, 0.5, 0.5), soit le gris moyen.

La même couleur extrapolée dans l'espace HSL sera :
C3 = (210, 1, 0.5), soit un bleu vert saturé.

En règle générale, les calculs de couleur, qu'il s'agisse d'extrapolation entre plusieurs valeurs ou de rendu dans le cadre d'images 3D, sont plus souples et visuellement plus prévisibles dans l'espace RGB, puisqu'ils n'introduisent pas de nouvelles teintes intermédiaires.

RGB, CMY, CIE et les autres

Il est évident que les trois modèles RGB, CMY et HSL peuvent être considérés comme des variantes du modèle de base XYZ du CIE, et de son dérivé pratique xyY. Paralèllement à ces approches, quasi universelles en infographie, plusieurs autres modèles sont mis en œuvre dans le domaine du traitement d'image, mais aussi dès lors qu'il s'agit d'interfaçage entre images numériques et télévision en particulier.
Ce dernier cas utilise fréquemment l'espace des couleurs du NTSC, le comité de normalisation de la TV américaine. Le modèle proposé, baptisé YIQ, est le fruit de l'accouplement du modèle CIE avec les contraintes techniques de bande passante (0 à 6 MHz) induites par la technologie de la télévision. A l'instar de son homologue CIE, la coordonnée Y contient toute l'information de luminosité de l'image. Elle est la seule exploitée par un téléviseur noir et blanc. Par ailleurs, le blanc de référence n'est autre que la source C préconisée par le CIE, quoique les informaticiens utilisent souvent le blanc D6500 sans que cela n'entraîne trop de différence.
Le codage de la teinte et de la saturation, quant à lui, fait appel aux composantes cyan-orange (I) et magenta-bleu (Q). La conversion YIQ à RGB ne fait appel qu'à une simple matrice constante. Les puristes pourront facilement la recalculer via l'espace xyY du CIE s'ils désirent répercuter la différence de référence du blanc entre la source C du NTSC et la source D6500 usuellement de mise pour les moniteurs couleur.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0,299 & 0,587 & 0,114 \\ 0,596 & -0,274 & -0,322 \\ 0,211 & -0,522 & 0,311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0,956 & 0,623 \\ 1 & -0,272 & -0,648 \\ 1 & -1,105 & 0,705 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

Le CIE, toujours lui, a tenté de mettre en place d'autres modèles de quantification de la couleur, basés non plus sur la mesure absolue des caractéristiques d'une couleur, mais plutôt sur la notion de différences entre couleurs, telles que nous les percevons. Pour ces modèles baptisés CIELAB et CIELUV, deux différences de couleurs perçues comme égales par le système visuel humain correspondent à une distance euclidienne constante dans ces espaces. Pour le système CIELAB, les coordonnées d'une couleur sont référencées par les paramètres L*, a* et b*. L*, commun aux deux modèles, est proportionnel à la racine cubique de la luminance.
La conversion entre CIELAB ou CIELUV est assez complexe. Par exemple, le passage de coordonnées XYZ CIE en CIELAB correspond généralement aux formules suivantes :

$$A = \frac{L^* + 16}{116}$$

$$X = Xn \left(\frac{a^*}{500} + A \right)^3$$

$$Y = YnA^3$$

$$Z = Zn \left(\frac{-b^*}{200} + A \right)^3$$

Simulation de la couleur

La discussion précédente concernant la modélisation des couleurs s'applique très bien à l'informatique, pourvu que les capacités de visualisation de l'ensemble "carte graphique plus moniteur" soient aptes à générer les 350 000 couleurs du spectre visible par le système visuel humain. Compte tenu du format général des données informatiques, la quantité numérique minimale se monte à trois octets ou 24 bits. On parvient théoriquement à représenter ainsi 256 x 256 x 256 = 16,7 millions de couleurs, par le biais de trois composantes RGB par exemple.
Malheureusement, les cartes graphiques couleur 24 ou 32 bits ne sont pas encore très répandues. Aussi, ce hiatus entre la généralisation des modèles couleur 24 bits et l'inadéquation des cartes graphiques usuelles, a-t-il exigé le développement d'algorithmes destinés à simuler au mieux une image 24 bits, sur un périphérique plus "pauvre" en termes de gestion de la couleur. Initialement développés au bénéfice des imprimantes monochromes, ils peuvent être généralisés aux systèmes de visualisation couleur. De ce fait, nous serons amenés à discuter des quatre grandes classes de méthode numérique de simulation des demi-teintes, avant d'en aborder la généralisation à la simulation de la couleur.

Le super-point, efficace mais dispendieux

La méthode la plus simple, et qui reste encore d'actualité pour ce qui concerne les périphériques monochromes haute résolution tels que les photocomposeuses, consiste à simuler numériquement le procédé classique des trames de similigravure. C'est à Holaday que l'on doit les premières implémentations de ce type d'algorithme. Le principe en est simple, puisqu'il s'agit simplement de faire correspondre à chaque point natif de l'image, une matrice de points du périphérique. Il suffira alors de remplir plus ou moins la matrice pour simuler un point natif plus ou moins dense. Examinons le cas d'une impression PostScript sur photocomposeuse laser. A la résolution de 2 400 dpi, et pour une linéature (fréquence) de trame de 150 dpi, c'est-à-dire une densité linéaire de 150 points natifs de l'image par pouce, on calcule aisément que pour chaque point natif, on dispose de 2 400/150 = 16 points du périphérique. Dans la mesure où la résolution du périphérique est uniforme dans les deux dimensions, à chaque point réel de l'image (natif) correspond une matrice de 16 x 16 = 256 points du périphérique. Grâce à un algorithme du type Holaday, il devient ainsi possible de simuler 256 valeurs de gris par point avec une résolution apparente de 150 dpi, suffisante pour que l'oeil puisse avoir une impression globale des niveaux de gris sur le film ou le bromure.
Pour simple qu'il soit, le procédé de trame simili numérique exige de hautes résolutions pour être efficace. Le même raisonnement, appliqué à une imprimante laser 300 dpi, implique que la simulation de 16 niveaux de gris fait chuter la résolution apparente à 75 dpi, et à 37,5 dpi pour 64 niveaux de gris, d'où une image très grossière et inacceptable.
Notons que les calculs précédents démontrent qu'une bonne

simulation des couleurs par la création numérique de films quadrichromiques exige une résolution minimale de 2 400 dpi au niveau de la photocomposeuse, et pour des linéatures de trames se situant aux alentours des 150 dpi. Si cette condition n'était pas respectée, certaines parties de l'image, notamment les grandes surfaces dégradées, laisseraient apparaître des bandes du plus mauvais effet, voire un appauvrissement des contrastes pour une image de type photographique.
Deux autres techniques sont donc apparues pour simuler des demi-teintes sans trop attenter à la définition finale de l'image. Techniquement il s'agit d'algorithmes de diffusion de l'erreur, dont le plus courant a été établi par Floyd et Steinberg, et d'algorithmes de tramage ordonné issus des recherches de Limb et Bayer. Une quatrième catégorie existe, qui tente de combiner les caractéristiques du tramage ordonné et de la répartition de l'erreur.

Diffusion de l'erreur

Considérons une image numérisée en 256 niveaux de gris. Chaque point est donc représenté par une valeur sur un octet (de 0 à 255). Pour une sortie sur imprimante, 0 correspond au blanc et 255 au noir. Le but est donc de convertir cette image en valeurs binaires 0 ou 1 compatibles avec la technologie de l'imprimante. Une première idée consiste à effectuer un effet de seuil par rapport à une intensité moyenne. Soit P(x,y) l'intensité d'un point quelconque de l'image repéré par ses coordonnées colonne, ligne :

```
si P(x,y)  127
  P'(x,y) = NOIR (1)
sinon
  P'(x,y) = BLANC (0)

  ERREUR = P(x,y) - P'(x,y)*255

P'(x,y) correspondant au point du périphérique, noir ou
blanc.
```

Simple et rapide, l'effet de seuil n'est pourtant pas très efficace puisque la perte d'information, et conjointement la perte des détails, est importante. Pour chaque point, il se produit une erreur d'estimation que l'on peut évaluer de la manière suivante :

*ERREUR = P(x,y) - P'(x,y)*255*

L'idée de Floyd et Steinberg est de répercuter cette erreur sur les plus proches voisins, plutôt que de la négliger purement et simplement. Dans une première implantation, les auteurs suggérèrent la répartition :

	*P(x+1,y-1)+3/16*ERREUR*
P(x,y)	*P(x+1,y)+5/16*ERREUR*
*P(x,y+1)+7/16*ERREUR*	*P(x+1,y+1)+1/16*ERREUR*

On s'aperçoit que ces spécifications entraînent une certaine complication pour l'écriture d'une routine de diffusion de l'erreur, car il est nécessaire d'évaluer des points situés sur la ligne précédente. C'est pourquoi un grand nombre d'implémentations utilisent une variante de l'algorithme original qui répartit l'erreur de gauche à droite et de haut en bas.

P(x,y)	*P(x+1,y)+3/8*ERREUR*
*P(x,y+1)+3/8*ERREUR*	*P(x+1,y+1)+2/8*ERREUR*

Les résultats obtenus sont excellents, mais aisément reconnaissables par l'aspect granuleux de l'épreuve.

Tramage ordonné

Le tramage ordonné, pour sa part, fait appel à un motif répétitif qui permet de quantifier l'erreur d'approximation à une position donnée de l'image. Il s'agit en fait d'une matrice contenant tous les niveaux d'intensité intermédiaire qui se superposeraient à l'image. Si l'intensité d'un point de l'image est supérieure à la valeur de la matrice correspondant à sa position, il apparaîtra noir. Il s'agit en quelque sorte d'une combinaison du procédé de matrice de simili-gravure numérique, et d'un algorithme de diffusion de l'erreur. Par rapport à ce dernier, la répartition de l'erreur est constante pour une position donnée.
La conformation de la matrice est importante, car il s'agit de minimiser les motifs "déplaisants". En fait, il s'agit d'obtenir une matrice pour laquelle chaque point de faible intensité est environné par un point de forte intensité. Les matrices les plus adéquates sont dues à Limb et Bayer. D'autre part, la taille de la matrice induit directement le nombre d'intensités que l'on peut simuler : 4 x 4 = 16, 8 x 8 = 64, etc. La plus petite

```
       +      +
       | 0  2 |
D2 =   | 3  1 |
       +      +
```

matrice, permettant de simuler 4 niveaux d'intensité, s'écrit :
On déduit simplement les matrices d'ordre supérieur par la relation récursive suivante :

$$Dn = \begin{vmatrix} 4Dn/2 & 4Dn/2 + 2Un/2 \\ 4Dn/2 + 3Un/2 & 4Dn/2 + Un/2 \end{vmatrix}$$

où "Un" représente la matrice n x n dont toutes les valeurs sont 1. De ces équations, nous obtenons facilement la matrice

$$D4 = \begin{vmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{vmatrix}$$

4 x 4 permettant la simulation de 16 couleurs :
Selon le nombre C de couleurs à simuler, la matrice de tramage sera d'ordre n, avec n x n = C. L'algorithme de tramage ordonné est extrêmement simple et suit la logique suivante.
Soit l'intensité d'un point P(x,y) représenté par ses coordonnées colonne (x) et ligne (y) :
calcul de la position relative du point au sein de la matrice de tramage.

i = (x modulo n) + 1
j = (y modulo n) + 1

si P(x,y) > Dn(i,j)
P'(x,y) = NOIR (1)
sinon
P'(x,y) = BLANC (0)

Un algorithme de tramage ordonné s'avère généralement très efficace, tant du point de vue du rendu que de celui de la vitesse d'exécution. Le rendu d'une telle méthode se reconnaît à la représentation parfois très géométrique de l'image qui est constituée par la superposition d'un grand nombre de croix.

Les techniques mixtes

Si les deux méthodes précédentes sont actuellement les plus couramment mises en oeuvre, Knuth en 1987 a été parmi les premiers à proposer une méthode mixte combinant à la fois le tramage ordonné et la diffusion de l'erreur. Sans développer la technique, remarquons que dans l'algorithme précédent, la comparaison entre un point et sa valeur relative dans la matrice induit un effet de seuil variable en fonction des coordonnées du point. Or, qui dit seuillage, dit également erreur que l'on peut estimer de la même manière que lors de notre discussion de l'algorithme de Floyd-Steinberg. Il suffit alors de la répartir sur les pixels avoisinants pour obtenir un croisement entre les deux méthodes. Bien entendu, une telle approche entraîne un nombre de calculs plus important, mais en contrepartie, le rendu peut être saisissant.
Chacun de ces algorithmes peut lui-même être combiné avec des filtres de traitement d'images classiques. C'est ainsi que nombre de méthodes de simulation de couleurs emploient une technique de réhaussement des contours à l'aide d'une convolution simple. Pratiquement, au lieu de tester l'intensité du point P(x,y), il suffit de calculer la résultante de la

$$\begin{vmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{vmatrix}$$

matrice de convolution :
Soit P'(x,y) = 9 P(x,y) - somme(8 plus proches voisins).
L'utilisation d'un filtre de convolution de ce type donnera à l'image résultante un meilleur "piqué" dans la mesure où les contours seront mieux définis. Partant de là, il est tout à fait envisageable de coupler une méthode de simulation des couleurs avec une convolution simple quelconque. On pensera notamment à une opération d'anti-aliasing.

Généralisation à un affichage 256 couleurs

La tendance actuelle de la technologie informatique usuelle est au support des modèles couleur dits réalistes, 24 ou 32 bits par point. Toutefois, la majorité des cartes graphiques courantes sont des dispositifs 8 bits, capables de visualiser 256 couleurs à partir d'une palette de 16,7 millions. Cette tendance ira d'ailleurs croissant avec la généralisation d'environnements graphiques du type Windows ou MacOS. Le cas particulier des cartes VGA étendues pose un léger problème, car les composantes RGB de chaque couleur ne sont effectivement codées que sur 6 bits, soit une palette réelle d'environ 262 000 couleurs. Toutefois, la logique de programmation des cartes graphiques reste 8 bits. Ainsi, toutes les adaptations des algorithmes de simulation de demi-teintes resteront-elles acceptables dans ce cas particulier important. Une approche pratique du problème consiste à considérer chaque composante comme une entité sur la-

LA GESTION DE LA COULEUR

Divers procédés de rendu couleur :
a) original 24 bits

b) palette optimisée 256 couleurs, tramage par diffusion de l'erreur

quelle opèrera indépendamment la logique de conversion de l'algorithme. Il est possible d'ajuster la palette des 256 couleurs disponibles de manière optimale afin de visualiser une image donnée. Une rose, par exemple, se contentera de 256 nuances de rouge, avec un résultat très réaliste à l'écran. Mais il s'agit là d'une méthode qui n'est pas universelle, car elle dépend trop des caractéristiques colorimétriques de l'image, et surtout, elle entraîne une analyse préalable de celle-ci, processus qui peut s'avérer fort dispendieux.
Il est plus valable d'appliquer nos raisonnements généraux aux cartes graphiques. En d'autres termes, nous allons créer, à l'aide de la LUT de la carte, un espace des couleurs RGB traditionnel, quoique plus limité. Chaque point étant codé sur 8 bits, il est tout à fait envisageable de consacrer à chaque primaire RGB un nombre de bits fixe : 3 pour le rouge, 3 pour le vert, et 2 pour le bleu. Le choix du bleu pour la composante 2 bits restante est fait en fonction des caractéristiques de l'oeil humain, qui discerne moins de différences dans cette partie du spectre. Il devient alors facile de charger une palette (LUT) universelle 332 constante, avec par exemple la convention suivante :

bit 0-1 composante bleue (4 niveaux)
bit 2-4 composante verte (8 niveaux)
bit 5-7 composante rouge (8 niveaux)

On obtient ainsi une correspondance exacte, quoiqu'appauvrie, entre la composition d'une couleur et son emplacement dans la LUT de la carte graphique.
Une première conversion 24 bits à 8 bits paraît évidente. Soit r,g,b les trois composantes, chacune sur 8 bits. La conversion sera de la forme :

r'= r/32 g'=g/32 b'=b/64.

Une fonction C opérant cette conversion pourrait prendre la forme :

```
RGB_to_Index (unsigned char r, unsigned char g, unsigned
char b)
{
  return( (r>5<5) + (g>5<2) + (b>6));
}
```

L'utilisation des instructions de décalage montre plus clairement la logique de la fonction. Une optimisation évidente consisterait à faire appel à des masques de bits et à l'opérateur C "&" (ET logique).
Cette conversion élémentaire est tout à fait comparable à la méthode du seuillage simple que nous avons précédemment évoquée. Elle induit la même perte d'information. Sur certaines images numérisées le rendu est acceptable, mais en règle générale, l'emploi de cet algorithme, certes rapide, induit des discontinuités de couleur inacceptables. Le rendu des surfaces dégradées, notamment, s'avère tout à fait incorrect.
Aussi allons-nous utiliser une méthode de tramage ordonné appliqué à la simulation des couleurs 24 bits. La première étape consiste bien sûr à établir la matrice correcte capable de discriminer 256 niveaux d'intensité. Elle est déduite des matrices de Limb et de la relation récursive précédente. Dans le cadre d'une implantation en langage C, elle prend la forme suivante :

```
unsigned char matrix16x16 [16][16]={
    {0,128,32,160,8,136,40,168,2,130,34,162,10,138,42,170},
    {192,64,224,96,200,72,232,104,194,66,226,98,202,74,234,106},
    {48,176,16,144,56,184,24,152,50,178,18,146,58,186,26,154},
    {240,112,208,80,248,120,216,88,242,114,210,82,250,122,218,90},
    {12,140,44,172,4,132,36,164,14,142,46,174,6,134,38,166},
    {204,76,236,108,196,68,228,100,206,78,238,110,198,70,230,102},
    {60,188,28,156,52,180,20,148,62,190,30,158,54,182,22,150},
    {252,124,220,92,244,116,212,84,254,126,222,94,246,118,214,86},
    {3,131,35,163,11,139,43,171,1,129,33,161,9,137,41,169},
    {195,67,227,99,203,75,235,107,193,65,225,97,201,73,233,105},
```

c) palette fixe 332, tramage ordonné

d) palette 8 couleurs, tramage ordonné

```
        {51,179,19,147,59,187,27,155,49,177,17,145,57,185,25,153},
        {243,115,211,83,251,123,219,91,241,113,209,81,249,121,217,89},
        {15,143,47,175,7,135,39,167,13,141,45,173,5,133,37,165},
        {207,79,239,111,199,71,231,103,205,77,237,109,197,69,229,101},
        {63,181,31,159,55,183,23,151,61,189,29,157,53,181,21,149},
        {255,127,223,95,247,119,215,87,253,125,221,93,255,117,213,85},
                };

/* conversion compensée RGB en un point 332          */
/* nécessite les coordonnées x et y d'un point       */
/* ainsi que ses composantes r,g,b 24 bits           */
/* renvoie l'indice de la LUT correspondant au mieux */

int Trame_Ordonnee(int x,int y,int r,int g,int b)
{

/*calcul de la position relative au sein de la matrice*/
  x%=16;y%=16;

                if( r  matrix16x16 [x][y] ) r=0;

                if( g  matrix16x16 [x][y] ) g=0;

                if( b  matrix16x16 [x][y] ) b=0;


                return((r>5<5)+(g>5<2)+(b>6));
}
```

Par ce biais, une surface de couleur unie sera représentée par un ensemble de points de couleur proche qui engendrera une impression générale plus nuancée. Le résultat est bien plus réaliste que dans le cas d'un seuillage simple. Cet exemple, tout à fait exploitable, donne le cadre général d'implantation de routines de simulation de la couleur. L'une des améliorations simples, qui fournit d'ailleurs d'excellents résultats visuels, consiste à moduler la valeur résultante de la composante couleur après le seuillage. Ici, nous l'avons purement et simplement annulée en cas de test négatif. Il est facile de modifier la routine afin de calculer cette valeur résultante en fonction de la différence entre la valeur de la composante et la valeur de seuil de la matrice. L'introduction d'un facteur plus ou moins aléatoire donne également d'excellents résultats. Un exemple possible pourrait prendre la forme suivante :

```
/*conversion compensée RGB en un point 332*/
/*avec introduction d'un facteur pseudo-aléatoire*/
int Trame_Ordonnée(int x,int y,int r,int g,int b)
{
  static int plusoumoins=1;
  x%=16;y%=16;

                if( r  matrix16x16 [x][y] && r9 && r)
                    { r=r+9*plusoumoins;plusoumoins*=-1;}
                if( g  matrix16x16 [x][y] && g9 && g)
                    {g=g+9*plusoumoins;plusoumoins*=-1;}
                if( b  matrix16x16 [x][y] && b21 && b)
                    {b=b+21*plusoumoins;plusoumoins*=-1;}
                plusoumoins*=-1;

                return((r>5<5)+(g>5<2)+(b>6));

}
```

Ici, le seuillage introduit un accroissement ou une diminution constante de la composante en cas de test négatif. La modification de la constante introduit un effet plus ou moins spectaculaire. Par ailleurs, l'introduction d'une variable bascule pour chaque composante favorise l'action de l'algorithme. Quoi qu'il en soit, les variations sur ce thème sont quasiment illimitées, et le rendu d'une image 24 bits par le biais d'une carte 256 couleurs peut être assez étonnant. L'implantation d'autres méthodes de simulation, notamment celle de Floyd Steinberg, suit le même caneva général, et ne pose pas de problème particulier. □

OBJET ET BASES DE DONNÉES

NENAD CETKOVIC

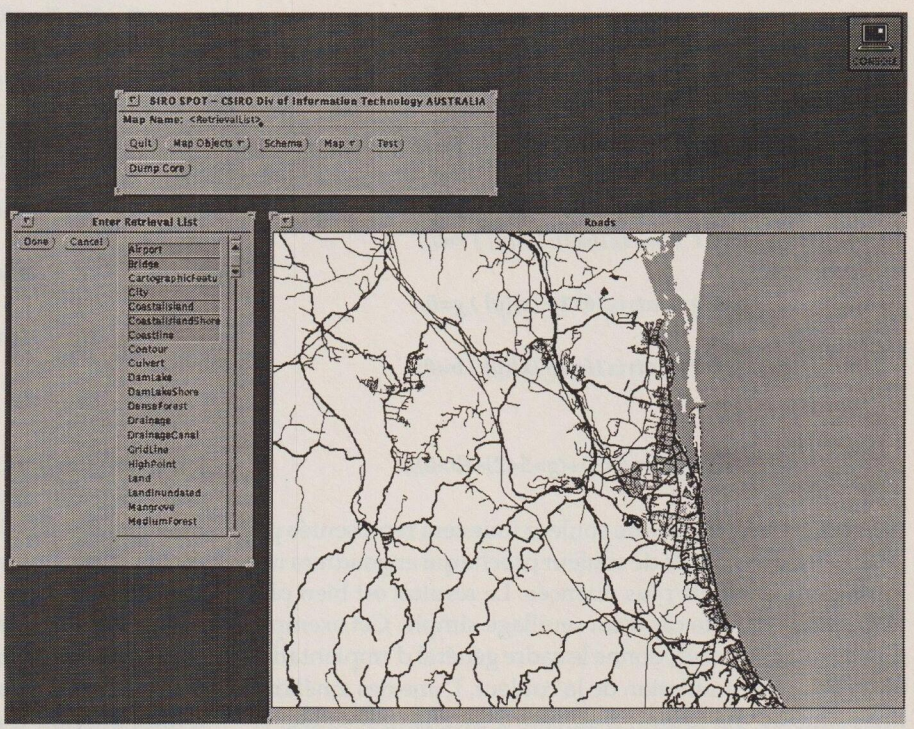
Trois grands domaines d'applications se dégagent de l'étude détaillée des besoins en base de données : la gestion, les données scientifiques, et enfin les données dites techniques comprenant notamment l'ensemble des besoins en génie logiciel. Le domaine de la gestion a incontestablement été le moteur de toutes les évolutions : l'utilisation d'un langage comme Cobol, dans les années 70, qui implémente les notions de structures de données (ce que ne faisait pas Fortran à l'époque), a très naturellement contribué aux développements successifs des modèles hiérarchique puis CODASSYL. Ensuite, l'évolution vers les systèmes de gestion de bases de données relationnelles n'a elle-même été qu'une amélioration essentiellement destinée aux mêmes besoins.

Les limites du relationnel

Si les avantages de la méthode relationnelle sont nombreux, il n'en demeure pas moins que cette technique présente encore bon nombre de limitations : les SGBDR ne sont en aucun cas adaptés pour des applications «temps réel» qui exploiteraient les données. De plus, ces bases sont encore trop souvent conçues dans une optique d'édition de données plutôt que de traitement ou d'aide à la décision. Enfin, d'un point de vue purement technique, l'organisation d'un SGBDR s'avère n'être qu'un empilement de fichiers à plat («flat files»). Aucune structure n'est conservée dans la sauvegarde des données : c'est à l'application de préparer la structure des données pour la sauvegarde puis, lors du chargement, de les importer et de les organiser dans les structures mémoire existantes qui les utiliseront. La conséquence est simple : il n'existe donc aucun rapport entre le stockage sur disque des données et les structures utilisées en mémoire.

La principale lacune d'une base de données conventionnelle réside dans le fait que les structures de données conventionnelles sont figées. En effet, le plus perfectionné des systèmes de gestion de données ne saura faire face à un changement dans la structure des données - notamment dans le cadre d'un enrichissement de ces dernières. La seule solution disponible, à laquelle tous les utilisateurs ont recours, consiste à redéfinir la structure existante puis à réorganiser la base, opération ô combien coûteuse en temps et en moyens humains. De plus, cette méthode exige la cessation momentanée de l'exploitation des données. Au final, cette solution n'est en fait qu'«un bol d'air» pour l'utilisateur puisqu'à la moindre modification, l'opération devra être effectuée à nouveau.

Un problème similaire d'évolution de la structure des données s'était déjà posé dans le monde du génie logiciel. La solution naturelle à ce problème porte un nom : les langages orientés objets. D'où l'idée qui a germé, dans l'esprit de bon nombre de chercheurs, d'intégrer la notion d'objet au domaine des bases

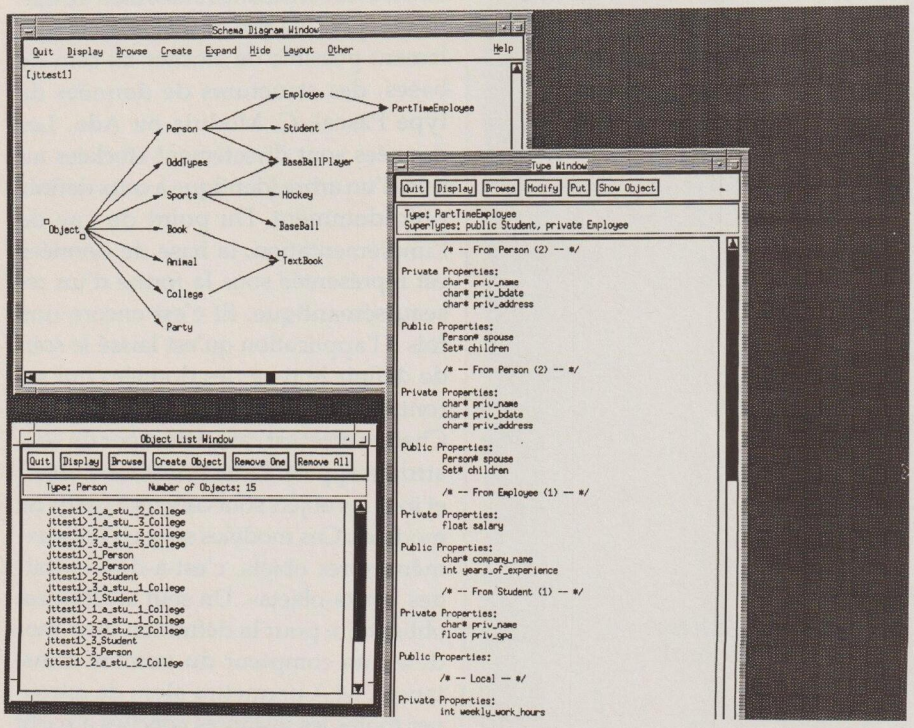


Exemple d'application d'Ontos.

Si le modèle relationnel s'est rapidement imposé face au modèle hiérarchique, l'évolution vers la notion d'objet correspond moins à une volonté d'imposer un nouveau standard qu'au souci de répondre à des besoins industriels qui n'ont pas encore trouvé de solution simple.

de données. M.A, de la société S.Y.R, et Ontos, commercialisé par BIM Systèmes, sont deux produits de gestion de données orientés objets, actuellement disponibles sur le marché français. Si les deux produits se classent dans la même famille d'applications, leurs approches sont extrêmement différentes : Ontos est un produit conçu pour fonctionner en harmonie avec le langage C++. De plus, il est déjà présent sur bon nombre de systèmes (la version Windows 3.0 est prévue avant la fin du premier semestre). La notion est des plus simples puisque le produit intègre les principales notions du langage C++. Quant à M.A, développé en langage ADA, il s'agit d'un produit que l'on pourrait qualifier de «tout-objet», aux concepts beaucoup plus complexes. Il nécessite néanmoins, avant une commercialisation «grand public», quelques travaux de finition. C'est notamment vrai pour ce qui concerne les outils d'interrogation qui ne sont absolument pas finalisés.

Expliquer dans son ensemble l'implémentation des objets et leurs fonctionnalités n'est pas le but du présent article. En effet, le magazine tout entier n'y suffirait probablement pas pour un explicatif complet. Il s'agit plutôt de décrire les notions fondamentales des bases de données orientées objet. En effet, à l'heure actuelle, cette technique est peu connue et seul un petit nombre d'informaticiens développent véritablement sous ce genre de système.



Structure intérieure d'une base orientée Objet.

L'attrait de l'objet

Si les produits diffèrent dans leur concept, un certain nombre de fonctionnalités communes pourraient constituer une définition formelle des fonctions d'une base de données objet. Première fonctionnalité, la persistance des objets fait que ces derniers «survivent» à la fin d'une session et sont exploitables lors de la session suivante. De plus, ces objets sont manipulables en groupe et peuvent être répartis en différents serveurs. Cette distribution des données doit toutefois être transparente, pour l'application bien sûr mais également pour l'utilisateur, et elles doivent être accessibles simultanément par différentes applications. Du point de vue de l'utilisateur, l'accès aux données est facilité par l'existence d'«opérateurs navigationnels» permettant de se déplacer à travers les différentes relations. Avec ces fonctions, les bases de données orientées objet prennent véritablement toute leur envergure. En effet, la méthode conventionnelle ne permet la «navigation» au sein des relations que sous la forme de jointures entre les différentes tables. Dans le modèle objet, cette même opération devient tout à fait naturelle et contraste avec la complexité du modèle relationnel.

L'optique dans laquelle a été conçu le produit M.A est celle d'une utilisation en milieu industriel (automatisme, CAO, électronique, etc.).

Dans de tels cas de figure, il est néces-

saire de pouvoir modéliser très rapidement une machine outil, un photocopieur ou, plus globalement, n'importe quel engin en cours de production. Les bases de données contiennent dès lors des milliers, voire des millions, de composants de base servant à la réalisation du projet. Chacun de ces composants a ses propres caractéristiques, des comportements typiques lorsqu'il est soumis à des stimuli externes et est, de plus, susceptible d'évoluer. On comprend dès lors que concevoir une base de données de ce type avec un SGBD, fut-il relationnel, relève de la prouesse intellectuelle.

Le principe de base de la gestion objet, dans ce cas précis, réside dans la notion d'arbre de taille et de forme variables. De fait, cet arbre est structuré comme un arbre lexicographique où les séparateurs, représentant les branches, sont entièrement paramétrables par l'application. Alors qu'un SGBDR permet de naviguer au sein de tables à l'aide de fonctions simples (enregistrement précédent ou suivant, critère de recherche, fin de fichier, etc.), dans le cadre d'une base objet il existe un ensemble de fonctions qui permettent de naviguer dans un arbre.

En pratique, il est possible de se promener dans n'importe quel sens dans un arbre : descente dans l'arborescence, remontée, mais aussi parcours de l'arbre sur un même niveau. De plus, si la première recherche prend un peu de temps, tout rappel lié à cette recherche est instantané : en effet, un pointeur persistant est laissé sur la feuille de l'arbre en question et M.A gère constamment l'ensemble des pointeurs créés, à partir de n'importe quelle requête.

Autre caractéristique, la notion d'attribut n'a pas de sens pour le système de gestion de bases de données : les données que l'on y stockera peuvent dès lors être un entier, un texte, un objet, un agglomérat d'objets, voire même des structures construites aussi bien de manière répétitive que récursive. Il est également possible de stocker au sein de bases, des structures de données de type Pascal, C, Modula ou Ada. Les données sont directement stockées au sein d'un arbre identique à ceux définis précédemment. Du point de vue de l'implémentation, la base de données est représentée sous la forme d'un réseau sémantique. Et c'est encore une fois à l'application qu'est laissé le soin de définir le type des données qui seront structurées au sein de l'arbre.

Chaque objet est caractérisé par un seul attribut appelé "l'identificateur d'objet" et tous les objets sont conçus à partir de modèles. Ces modèles sont en fait eux-mêmes des objets, c'est-à-dire en fait des «méta-objets». Un seul attribut est obligatoire pour la définition d'un modèle : un compteur du nombre d'instances. M.A permettra alors de retrouver toutes les instances conçues à partir de cette définition (et vice versa !).

Le principe des requêtes est extrême-

EXEMPLE DE DÉFINITION D'UNE STRUCTURE C++ AU SEIN D'ONTOS

```
#include "reference.h"
#include "object.h"

class Set;

class Person : public Object
  {
  private:
     set*       privReferred;
     char*      privAddress;
        char*      privPhone;
     reference privReferredBy; // Person
  public:
     Person (APL *theAPL)
     Person ( char* name, char* address=(char*)NULL,
           char* phone=(char*)NULL,
                 char* referredBy=(char*)NULL);

     void Destroy (Boolean aborted = FALSE);
     void PutObject (Boolean deallocate = FALSE);
     void deleteObject (Boolean deallocate = FALSE);
     ~Person();

     Type *getDirectType();

     char* Phone();
     void Phone (char* phone);

     char* Address();
     void Address (char* address);

     Person* RefferedBy();
     void ReferredBy (Person* referredby);
     void ReferredBy (char* referredby);

     void InsertReferred (Person* referred) {
           if (referred) referred-ReferredBy (this);
           }
     Set* GetReferredSet();
     void displayData();
};
```

ment simple puisqu'il n'est pas nécessaire de connaître la structure des données pour formuler la requête. Une simple fonction SEARCH() implémente cette fonctionnalité. Parmi ses paramètres, le noeud courant de l'arbre M.A, la pile du système couramment utilisée et une ligne de commande. L'efficacité de cette méthode de recherche ainsi que la gestion des arbres sont tellement patentes qu'il est dès lors possible d'effectuer des traitements de données en temps réel avec ce système. Ce qui paraît inconcevable avec les bases de données actuelles.

Le stockage des données s'inspire également de la philosophie «tout objet» du produit : le système fonctionne sur un immense espace virtuel. En pratique, pour le développeur système, cet espace est la somme de tous les espaces disque disponibles. Les données y sont sauvegardées de manière à conserver la structure existant en mémoire. Il s'agit donc uniquement de copier l'objet sur le disque, et aucunement de le transformer par une quelconque opération de «mise à plat».

La conception de M.A n'est pas forcément facile à appréhender, tant elle diffère de la conception que l'on a habituellement d'une base de données. A l'inverse, toutes les bases de données orientées objets ne présentent pas systématiquement une telle complexité. Ontos, commercialisé par BIM Software, ne présente pas un aspect aussi «complexe» mais n'en demeure pas moins extrêmement efficace. Son credo est le suivant : implémenter, au sein des bases de données, une architecture entièrement objet. Dans un souci de pratique, le produit implémente également une interface avec des langages d'interrogation privilégiés.

Ontos, l'acquis de l'expérience

Si l'interfaçage d'Ontos est possible avec n'importe quel langage orienté objet (des versions avec Object Cobol sont par exemple en prévision), il est vrai que le langage C++ reste actuellement le langage objet préférentiel. Ce choix s'explique par le fait que C++, avec tous ses avantages et ses défauts, est devenu le standard de facto de l'industrie logicielle, dans le domaine des langages orientés objet. Aussi l'ensemble des objets et méthodes définis sous Ontos sont-ils interfaçables avec ce langage. Un exemple (fig. 1) illustre l'exportation d'une structure de la base vers C++.

L'implémentation des classes dans Ontos gravite autour d'un noyau principal, en l'occurrence une seule classe : la classe OBJECT. Toutes les classes créées sous Ontos héritent directement de cette classe, notamment pour deux propriétés : le «constructor» et le «destructor», deux méthodes dédiées à la création et à la destruction d'objets. Cette classe contient également l'implémentation des propriétés liées au nom des classes ainsi que les mécanismes de liens physiques des objets dans la base. De plus, les objets peuvent être structurés sous forme d'agrégats notamment grâce à des classes spécialisées comme les dictionnaires, «set», «array» et autres listes, structures conventionnelles de stockage des informations mais étendues ici aux notions d'objets.

L'accès aux supports de données (disque, sauvegarde etc.) se fait par un seul identifiant : l'identificateur associé à tout objet. Ce dernier est unique de manière à garantir l'unicité au sein d'une base. De plus, à la sauvegarde, l'objet

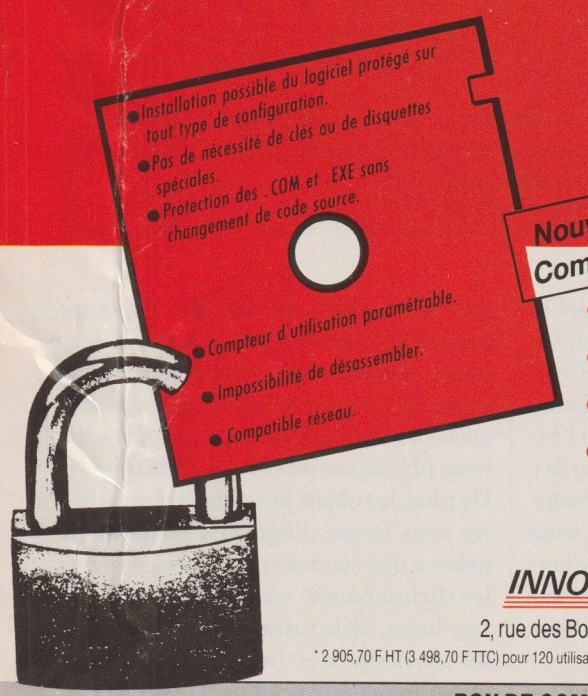
PETIT GLOSSAIRE DE L'OBJET

- **Classe**: Ensemble de «variables» et de propriétés servant de descripteur à la génération d'objets, ces derniers étant également appelés instances.
- **Généricité**: Notion permettant de paramétrer des classes d'objets. Il est dès lors possible de définir une classe qui sera instanciée en différentes autres classes par la précision du type des éléments. Par exemple, «Pile arithmétique d'entiers» ou «Pile arithmétique de caractères» sont deux classes générées à partir de la classe «Pile arithmétique» dont les éléments sont indéterminés.
- **Héritage**: Cette notion clé de la programmation objet permet de définir une classe B comportant des propriétés similaires à la classe A, uniquement en explicitant les innovations de la classe B. La représentation courante des classes se fait sous forme d'arborescence. La classe B viendra donc directement sous la classe A dans l'arbre des classes. L'héritage est ainsi répercuté de niveau en niveau, en descendant jusqu'aux feuilles.
- **Héritage multiple**: une classe peut hériter de plusieurs classes. Si l'on définit une classe "félidés", cette dernière peut être décrite comme héritant à la fois de la classe "mammifères" et de la classe "carnivore".
- **Instanciation**: Action consistant à construire un objet à partir d'une définition d'une classe. Par exemple, Toto est un objet créé à partir de la classe Person (c.f encadré en page xx).
- **Message**: Ordre d'activation d'une méthode.
- **Méthode**: Comportement typique des objets d'une classe. Si l'on fait appel à une méthode n'existant pas dans une classe, le système remonte l'arborescence des classes pour déclencher la première méthode correspondante qu'il rencontre.
- **Métaclasse**: toutes les classes d'un système ont une caractéristique commune : ce sont toutes des classes. Une classe au moins servira à la définition des propriétés de toutes les classes. Il s'agit de la métaclasse.

est la plus petite partie accessible par toute opération de lecture/sauvegarde. En effet, les différents constituants de l'objet sont physiquement liés à la sauvegarde.
La conséquence en est simple : la récupération sur disque de la structure ne demande qu'un seul accès disque. A noter que le même type de transaction dans un base relationnelle exigerait plusieurs accès consécutifs sur différentes tables, afin d'en extraire un objet similaire, c'est-à-dire l'objet pris dans sa globalité avec les diverses relations.

Relationsde type 1:N

On le voit, le grand avantage de la technique employée par Ontos consiste à représenter, au sein même de la structure objet, les relations de type 1:N directement. Lorsqu'on connaît l'importance vitale du traitement de ces relations, on comprend mieux l'attrait d'une telle méthode.
Construite autour d'une architecture client/serveur, la base de données exploite au mieux les capacités qui lui sont offertes notamment par un «lock» non pas au niveau de l'enregistrement, puisque cette notion n'existe plus, mais au niveau de l'objet.
Des fonctions primaires de traitement des objets sont également implémentées : Getobjet, GetlogicalCluster, GetClosure. Ces activations des objets effectuent simultanément tous les verrouillages nécessaires au niveau de l'objet, la base devant rester accessible aux accès concurrents.
De nombreux travaux sont en cours sur le thème des bases de données orientées objet. Ces dernières font l'objet de nombreuses recherches à l'heure actuelle, et l'on voit chaque année pointer de nouvelles implémentations. Les conférences de la dernière édition de Tools 90, le salon dédié au développement, traitaient encore de ce problème et les véritables implémentations grand public ne sont pas encore réalisées. Il n'en demeure pas moins que ces produits apparaissent sur un marché accessible à tous (OS/2, Unix, etc.) et s'adressent à des domaines d'applications qui ne disposaient pas jusqu'alors d'outils adaptés; il s'agit par exemple de la CAO, la géographie et en règle plus générale, de l'ensemble des gestions graphiques. Il est vrai que l'objet est, dans le domaine graphique, assez innovateur et important.
Après tout, c'est à des langages implémentant cette notion que l'on doit des environnements graphiques comme Windows 3.0. □



EVERLOCK, L'ANTIVOL LOGICIEL

Nouvelle version 2.1
Compatible Windows

- Haute protection par logiciel (contre Copy II PC, Copyright, etc.).
- Faible coût par copie.
- Entièrement paramètrable suivant vos besoins.
- Utilisation et installation faciles.

A PARTIR DE
2 950F HT*
(3 498,70F TTC)

Pour vous renseigner ou commander :

INNOSOFT (1) 45.06.76.91

2, rue des Bourrets 92150 SURESNES - FAX (1) 47.28.62.89

* 2 905,70 F HT (3 498,70 F TTC) pour 120 utilisation • 5 950 F HT (7 056,70 F TTC) pour 500 utilisations • 8 950 F HT (10 614,70 F TTC) pour la version illimitée. Documentation en français.

BON DE COMMANDE OU DEMANDE DE DOCUMENTATION

□ Je commande [_____] exemplaire(s) de EVERLOCK/120 utilisations à 3 563,93 F TTC (3 498,70 F TTC + 65,23 TTC de port)
□ Je commande [_____] exemplaire(s) de EVERLOCK/500 utilisations à 7 121,93 F TTC (7 056,70 F TTC + 65,23 TTC de port)
□ Je commande [_____] exemplaire(s) de EVERLOCK/illimité à 10 679,93 F TTC (10 614,70 F TTC + 65,23 TTC de port)
□ Je désire recevoir une disquette de démonstration et une documentation.

Ci-joint mon règlement :
□ Chèque
□ Carte Bleue Numéro : _____ Date d'expiration : _____
□ Contre-remboursement (60 F TTC de frais supplémentaires).

NOM : _____
SOCIÉTÉ : _____
ADRESSE : _____
CODE POSTAL : _____
VILLE : _____
A renvoyer à INNOFOST, 2, rue des Bourrets 92150 SURESNES

MS-DOS®

ET

TEMPS-REEL

AVEC INTEL, C'EST POSSIBLE

Le logiciel d'exploitation DOS/RMX d'Intel est le seul sur le marché à associer de vraies capacités DOS à de vraies capacités temps-réel multitâches.

Vous pouvez désormais offrir aux clients qui demandent des solutions à base de PC de véritables fonctionnalités temps-réel sans perdre la souplesse de DOS.

DOS/RMX a été développé par Intel, l'entreprise qui est à l'origine de la révolution micro-informatique. Cela signifie que vous êtes assuré d'une compatibilité totale avec tout le parc logiciel PC.

Il est possible d'écrire des extensions logicielles pour tirer profit des architectures 32 bits d'Intel. Ainsi, si vos clients souhaitent élargir les fonctionnalités disponibles en passant aux performances de niveau mini-ordinateur que l'on trouve sur un système MULTIBUS II, le logiciel DOS/RMX d'Intel demeure entièrement compatible, et les investissements logiciels de vos clients sont préservés.

Ajoutez à ces avantages la garantie de qualité des logiciels Intel, et vous disposez là d'une solution nouvelle sans équivalent.

Vous voulez voir une démonstration, vous avez besoin d'un complément d'information ? Contactez notre Département Communication : Intel, 1 rue Edison, BP 303, 78054 St Quentin en Yvelines cedex, Tél. (1) 30 57 70 00, poste 7101.

intel®

MS-DOS est une marque de Microsoft Corporation

TURBO VISION,

La bibliothèque objet du Turbo Pascal 6.0.

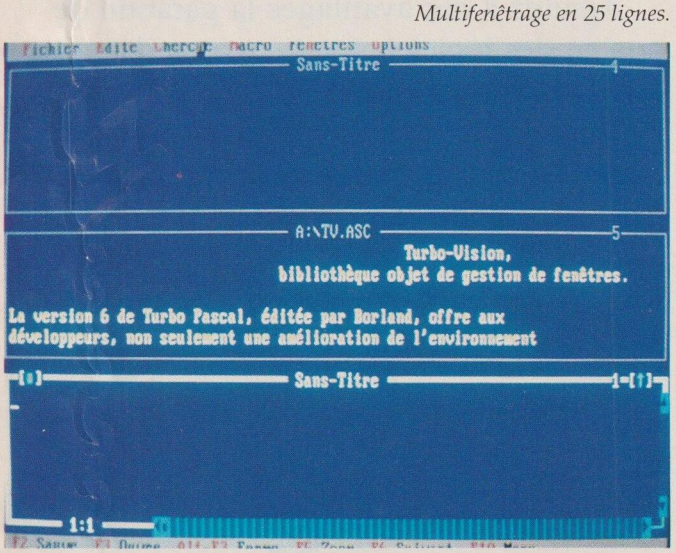
JEAN-PAUL PRUNIAUX

Avec cette bibliothèque, les programmeurs Pascal vont être amenés à raisonner différemment. La manière dont elle est organisée avec les méthodes de programmation événementielle, vont préparer plus d'un programmeur à une façon de penser qui leur sera utile pour Windows.
Pour prendre en main cet outil, nous avons choisi un des programmes de démonstration livrés avec Turbo Vision, et l'avons quelque peu retravaillé afin de lui ajouter certaines fonctionnalités : l'éditeur «TVEdit.Pas». Ceci nous a permis d'atteindre un double objectif en fabriquant aussi un outil de programmeur plus agréable.
Le présent article a été rédigé en supposant que le lecteur était déjà familiarisé avec les techniques de programmation objet; cela étant, il reste conseillé d'étudier le fichier «Editors.Doc», pour mieux comprendre le fonctionnement de l'objet «Editeur».

Nouveaux types de fenêtres

Pour se familiariser avec l'organisation des bibliothèques «Turbo Vision», on commencera par ajouter une table ASCII,

Multifenêtrage en 25 lignes.



style «SideKick», dans l'application. Cette opération peut être facilement réalisée car une unité d'objet «Table ASCII», «AsciiTab.Pas», fait partie des exemples livrés.
Il suffit de reprendre «TVEdit.Pas» (ou une copie, «TVE.PAS») et de faire les quelques retouches suivantes :
1)- Dans les déclarations d'unités, ajouter «AsciiTab», 2)- Dans la liste des commandes, ajouter la commande «cmAscii = ?», pour disposer d'une commande d'ouverture de fenêtre pour ces tables ASCII, 3)- Dans la méthode «TEditorApp.HandleEvent», traiter la commande «cmAscii» pour obtenir l'ouverture de cette fenêtre (une ligne). 4)- .Corriger les menus, afin de générer la commande «cmAscii».
Ces modifications trouvent leur illustration dans le listing ci-dessous :

```
{————————————————————}
{     Turbo Pascal
6.0                                          }
{     Turbo Vision Editor
Demo                                       }
{     Copyright (c) 1990 by Borland
International                          }

{
}
{ Extrait de modifications : Ajout de la table
ASCII                }
{                         Jean-Paul Pruniaux
10/1/91      }
{————————————————————}
```

```
program TVEdit;
{...}
uses DOS, Objects, Drivers, Memory, Views,
Menus, Dialogs,
  StdDlg, MsgBox, App, Calc, Buffers, Editors,
  AsciiTab;
{...}
const
  {...}
  cmCalculator = 104;
  cmShowClip   = 105;
  cmAscii      = 106;
```

La version 6 de Turbo Pascal, éditée par Borland, offre aux développeurs, non seulement une amélioration de l'environnement rejoignant l'éditeur multisource de Turbo C++, mais aussi une bibliothèque objet de gestion d'écran, de fenêtres, de souris... qui offre une interface utilisateur standard pour les développeurs Pascal.

```
{...}
procedure TEditorApp.HandleEvent(var Event:
TEvent);
{...}
  procedure Calculator;
     begin
     DeskTop^.Insert(ValidView(New(PCalculator,
     Init)));
     end;
  procedure Ascii;
     begin
     DeskTop^.Insert(ValidView(New(PAsciiChart,
     Init)));
     end;

  begin
  TApplication.HandleEvent(Event);
  case Event.What of
     evCommand:
        case Event.Command of
           {...}
           cmCalculator: Calculator;
           cmAscii : Ascii;
           {...}
        end;
  else
     Exit;
  end;
  ClearEvent(Event);
  end;

procedure TEditorApp.InitMenuBar;
  var
     R: TRect;
  begin
  GetExtent(R);
  R.B.Y := R.A.Y + 1;
  MenuBar := New(PMenuBar, Init(R, NewMenu(
     NewSubMenu('~F~ile', hcNoContext, NewMenu(
        {...}
  NewSubMenu('~E~dit',    hcNoContext, NewMenu(
        {...}
  NewSubMenu('~S~earch',  hcNoContext, NewMenu(
        {...}
  NewSubMenu('~W~indows', hcNoContext, NewMenu(
     NewItem('~S~ize/move', 'Ctrl-F5',
     kbCtrlF5, cmResize,
hcNoContext,
     NewItem('~Z~oom',   'F5',   kbF5,
     cmZoom,
hcNoContext,
     NewItem('~T~ile',   '',     kbNoKey,
     cmTile,
hcNoContext,
     NewItem('C~a~scade',   '',   kbNoKey,
     cmCascade,
hcNo Context,
     NewItem('~N~ext',   'F6',    kbF6,
     cmNext,
hcNoContext,
     NewItem('~P~revious','Shift-F6',
     bShiftF6, cmPrev
hcNoContext,
     NewItem('~C~lose',  'Alt-F3',  kbAltF3,
     cmClose,
hcNoContext,
     NewLine(
     NewItem('Ca~l~culator','',    kbNoKey,
cmCalculator,hcNoContext,
     NewItem('Asc~i~i',     '',    kbNoKey,
     cmAscii,
hcNoContext,
     nil))))))))))),
  nil)))))));
  end;
{...}
end.
```

Quel est le fonctionnement des objets de «Turbo Vision» ?

Le programme lui-même est basé sur un objet dérivé de «TApplication».
Le programme principal comprend 3 lignes du style :

```
begin
   Application.Init;
```

```
  Application.Run;
  Application.Done;
end.
```

Cet objet «TApplication» constitue l'enveloppe du programme; il comprend la description du menu, de la ligne d'état et de l'espace de travail (DeskTop). La méthode d'initialisation de l'application appelle les méthodes d'initialisation de ces trois objets. C'est généralement dans l'espace de travail que seront insérées les diverses fenêtres.

Turbo Vision» est un bibliothèque permettant le développement d'applications basées sur la programmation événementielle. Une structure de type «TEvent» est générée par une des méthodes de l'application (GetEvent). Cette boucle est appelée par «Application.Run».

«GetEvent» indique au moyen de cette structure ce qui vient de se passer : frappe de touche au clavier, déplacement de souris, etc. L'événement est ensuite acheminé vers les divers objets de l'application jusqu'à ce qu'il soit traité (ceci est signalé au moyen de la procédure «ClearEvent»). Chaque objet intégré dans l'application possède une méthode «HandleEvent» dont la charge consiste à traiter l'événement (déplacement du curseur, par exemple).

Au plus haut niveau, ces événements peuvent être interprétés pour fournir des codes de commandes afin de simplifier le travail d'interprétation des diverses routines «HandleEvent». La structure «TEvent» est alors de type «commande» et comprend l'un de ces codes.

Le listing ci-contre montre la création d'un code de commande «cmEvent»; lorsque ce code se présentera dans «Application.HandleEvent», nous ouvrirons une fenêtre «Table ASCII».

Dans la routine «Ascii» de «HandleEvent», on remarquera que la variable «DeskTop» contient l'adresse de l'objet «espace de travail». Cette routine crée une fenêtre («New»), teste le résultat de cette création («validView») et l'insère dans le DeskTop.Finalement, nous avons repris la méthode «InitMenuBar» pour insérer dans le sous-menu fenêtre une commande qui génère «cmAscii».

Mise au point de «Editors.Pas»

Après avoir compilé une première fois «TVEdit» et l'avoir testé, des problèmes de comportement se sont manifestés :

Le clavier numérique ne répond pas au «NumLock», certaines touches numériques (3 à 6) ne permettent pas d'introduire de chiffres, que ce soit depuis le clavier alphabétique ou numérique.

Ces problèmes auront probablement disparu avec la version française, mais leur examen s'est avéré être intéressant pour une bonne compréhension d'ensemble.

L'extrait du listing de «Editors.Pas» reproduit ci-contre met en évidence la manière dont sont interprétées les commandes dans cet objet.

```
{ Extrait de TVision\Editors.Pas                }
{ Unité et objet «fenêtre d'édition
de texte»                                       }
{ les ... indiquent des éléments de source non
listés                                          }
{-------------------------}

{ Constantes définissant les traitements à
effectuer}
const
  cmSave          = 80;
  cmSaveAs        = 81;
  cmFind          = 82;
{etc ...}

{ 3 tables convertissant des frappes de
touches en commandes }
const
  FirstKeys: array[0..37 * 2] of Word = (37,
     Ord(^A),       cmWordLeft,     Ord(^C),
     cmPageDown,
     Ord(^D),       cmCharRight,    Ord(^E),
     cmLineUp,
     Ord(^F),       cmWordRight,    Ord(^G),
     cmDelChar,
     {etc ...}
     kbDel,         cmDelChar,   kbShiftIns,
     cmPaste,
     kbShiftDel,    cmCut,          kbCtrlIns,
     cmCopy,
     kbCtrlDel,     cmClear);
  QuickKeys: array[0..8 * 2] of Word = (8,
     Ord('A'),      cmReplace,      Ord('C'),
     cmTextEnd,
     Ord('D'),      cmLineEnd,      Ord('F'),
     cmFind,
     Ord('H'),      cmDelStart,     Ord('R'),
     cmTextStart,
     Ord('S'),      cmLineStart,    Ord('Y'),
     cmDelEnd);
  BlockKeys: array[0..5 * 2] of Word = (5,
     Ord('B'),      cmStartSelect, Ord('C'),
     cmPaste,
     Ord('H'),      cmHideSelect,  Ord('K'),
     cmCopy,
     Ord('Y'),      cmCut);
  KeyMap: array[0..2] of Pointer = (@FirstKeys,
@QuickKeys, @BlockKeys);

{ Routine convertissant frappe de touche en
commande grâce aux tables}
{-------------------------}
 function ScanKeyMap(KeyMap: Pointer; KeyCode:
Word): Word; assembler;
asm
     PUSH    DS
     LDS     SI,KeyMap      {DS:SI = adresse
     de la table décodage  }
```

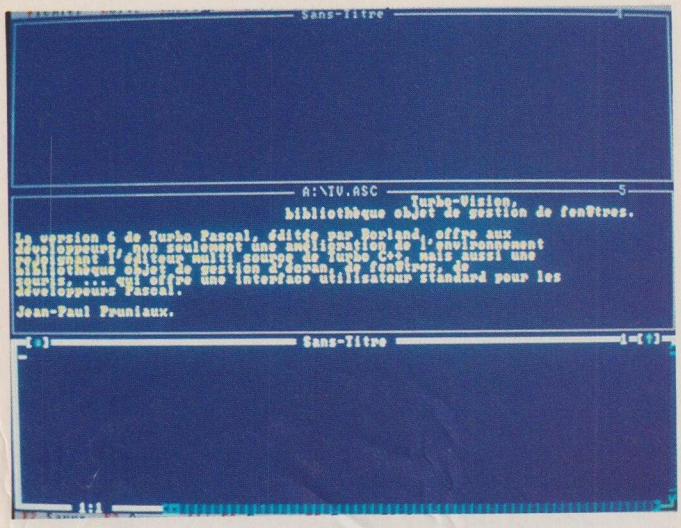
```
    MOV     DX,KeyCode      {DX = touche à
tester                  }
    CLD
    LODSW
    MOV     CX,AX           {CX = taille de la
table                   }
@@1: LODSW
    MOV     BX,AX           {Touche dans table :
Scan+Code et ASCII}
    LODSW                   {Commande
équivalente                 }
    OR      BL,BL           {Touche étendues
(ASCII = 0)
?           }
    JE      @@2             {Aller tester
1'égalité des codes     }
    CMP     BL,DL           {Le code ASCII
est-il le même          }
    JNE     @@3
    OR      BH,BH           {Ne pas comparer les
Scan-code               }
    JE      @@4
@@2:    CMP     BH,DH       {ERREUR :
L'instruction est CMP BX,DX  }
    JE      @@4             {Code trouvé,
commande renvoyée par AX }
@@3:    LOOP    @@1         {Examen code
suivant dans la table   }
    XOR     AX,AX           {Code non trouvé,
retour 0 par AX     }
@@4:    POP     DS
end;
```

```
{ Traitements des événements et commandes }
procedure TEditor.ConvertEvent(var Event:
TEvent);
var
  Key: Word;
begin
  if Event.What = evKeyDown then
  begin
    Key := Event.KeyCode;
    if KeyState <> 0 then
    begin
      if (Lo(Key)> = $01) and (Lo(Key) <= $1A)
then Inc(Key, $40);
      if (Lo(Key)> = $61) and (Lo(Key)<=  $7A)
then Dec(Key, $20);
    end;
    Key := ScanKeyMap(KeyMap[KeyState], Key);
    KeyState := 0;
    if Key <> 0 then
      if Hi(Key) = $FF then
      begin
        KeyState := Lo(Key);
        ClearEvent(Event);
      end else
      begin
        Event.What := evCommand;
        Event.Command := Key;
      end;
  end;
end;

procedure TEditor.HandleEvent(var Event:
TEvent);
var
  {...}

begin
  TView.HandleEvent(Event);
  ConvertEvent(Event);      {Traduction
des événements clavier    }
  {...}
  case Event.What of
    evMouseDown:
      begin
      {...}
      end;
    evKeyDown:
      case Event.CharCode of
      {...}
      end;
    evCommand:
      case Event.Command of
      {...}
        cmCut:   ClipCut;
        cmCopy:  ClipCopy;
        cmPaste: ClipPaste;
      {...}
      end;
    evBroadcast:
      case Event.Command of
      {...}
      end;
  end;
  ClearEvent(Event);
end;
```

Dans cette unité, l'objet «TEditWindow» est une fenêtre (type «TWindow») permettant l'édition d'un fichier ASCII. Celle-ci contient un objet «TFileEditor» dérivé de «TView» par l'intermédiaire de «TEditor», il s'agit de l'éditeur à proprement parler. Ces objets respectent l'organisation des éléments constitutifs de «Turbo Vision».

Le problème qui a été traité, ici, est celui de l'interprétation des touches de déplacement et des commandes compatibles «WordStar», auxquelles beaucoup de programmeurs sont habitués (<Ctrl><K>, <Ctrl><Q>)..

«*TEditor.HandleEvent*» reçoit les événements pr «TView.HandleEvent». Si ceux-ci concernent des frappes clavier, ils vont être éventuellement traduits pour fournir commandes (déplacement de curseur, par exemple). La-



Multifenêtrage en 43 lignes.

tine «ConvertEvent» se charge de ce travail d'examen et d'interprétation.
La manière dont ce travail doit être fait est mémorisé dans les tables «FirstKeys», «QuickKeys» et «BlockKeys». Chacune de ces tables contient le code d'une touche, suivi du numéro de commande qu'elle doit générer. Par défaut, c'est la table «FirstKey» qui sera utilisée; mais si <Ctrl><K> ou <Ctrl><Q> sont utilisés, c'est l'une des deux autres tables qui devra être prise en compte (ceci est mémorisé dans la première table par les codes «$FF01» et «$FF02»). Les adresses de ces tables sont mémorisées dans le tableau «KeyMap».
Un sous-programme en assembleur, «ScanKeyMap», examine une de ces tables (dont l'adresse est passée comme un paramètre) pour tenter d'y trouver le caractère à analyser (deuxième paramètre). S'il y parvient, «ScanKeyMap» renvoie le code de la commande associée, sinon «0».
C'est dans cette routine que résident les problèmes précédemment évoqués.
Pour comprendre de quoi il s'agit, il convient d'examiner comment sont codées les frappes de touches dans l'environnement «Turbo Vision». La structure «TEvent» comprend, entre autres, les champs suivants :

```
type
  TEvent = record
     What : Word;
     case Word of
        evKeyDown : (
           case Integer of
              0 : (KeyCode : Word);
              1 : (CharCode : Char;
                   ScanCode : Byte));
end;
```

«evKeyDown» est une constante «Turbo Vision» permettant d'interpréter la nature de l'événement : une frappe de touche; cette identification se fait au moyen du champ «TEvent.What».
«TEvent.KeyCode» récupère la frappe d'une touche telle qu'elle a été interprétée par le «BIOS» de la machine (interruption 16h). Ce champ est décomposé en deux sous-champs «TEvent.CharCode» et «TEvent.ScanCode» contenant, l'un le code ASCII du caractère, l'autre le code de balayage du clavier.
D'autre part, pour ainsi dire toutes les frappes de touches d'un clavier PC sont identifiées sous forme de constantes «kb???» dans «Turbo Vision»; ce sont ces valeurs que l'on retrouve dans «TEvent.KeyCode». Y figurent ainsi les combinaisons «<Shift><flèche>» (qui en fait n'existaient pas sur les premiers PC).
Dans nos tableaux «???Key», nous pouvons aussi bien trouver certaines de ces touches (l'octet équivalant à «CharCode» vaudra bien souvent «0») que des touches de contrôle («Ord(^A)» par exemple) dont l'équivalent du «ScanCode» vaudra toujours «0», ce qui n'est pas le cas de la valeur transmise.
Dans son travail d'interprétation, «ScanKeyMap» traite et corrige ses informations, malheureusement le test en ligne «@@2:» ne porte pas sur la valeur corrigée (OR BH,BH deux lignes avant). L'instruction corrigée qui procure un fonctionnement correct de l'éditeur est «@@2: CMP BX,DX».
Signalons ici quelques problèmes d'incompatibilité BIOS rencontrés au cours du travail :

- Sur certains systèmes, le driver de l'interruption 9 («KEYBFR.COM»), voire le «BIOS», ne sont pas 100% compatibles, le clavier numérique et le clavier étendu (flèches) sont alors confondus. Le clavier 102 touches réagit dès lors comme un 83 touches. Ceci se résout simplement en changeant de version de «KeybFr» pour en installer un qui soit vraiment compatible.
- Sur une machine, nous avons vu ce problème persister, malgré une gamme d'essai assez large; il s'agit probablement d'une incompatibilité «BIOS» au niveau du driver de l'interruption 16h. La solution ici consisterait à écrire un module résident qui redrive cette interruption 9. Ceci pourrait être réalisé en capturant le code de l'interruption 16h d'une machine saine grâce à un debugger. Malheureusement, le temps nous manquait pour ce faire.

Ajout de commandes à l'éditeur

Quitte à manipuler le driver d'événement de l'éditeur, nous en avons profité pour effectuer un certain nombre de modifications :

- Modification de la gestion des commandes <page Haut> et <page Bas> afin de conserver le curseur sur la même ligne d'écran après que le texte ait défilé (au lieu de le renvoyer

sur la première ou dernière ligne de la fenêtre),

■ Ajout des commandes <Ctrl Home> et <Ctrl Fin> pour envoyer le curseur en haut ou en bas de l'écran sans faire défiler le texte,

■ Gestion de tabulation logique (remplie avec des espaces) de largeur variable (paramètre),

■ Création d'une commande <Ctrl P> permettant d'insérer un caractère de contrôle dans le texte (commande style «WordStar»),

■ Deux nouvelles touches de gestion de bloc (F7 et F8),

■ Gestion de 9 marques permettant de mémoriser une position de texte et d'y retourner ultérieurement (<Ctrl><K> 1 à 9, puis <Ctrl><Q> 1 à 9).

Voici le listing des modifications apportées à cet éditeur :

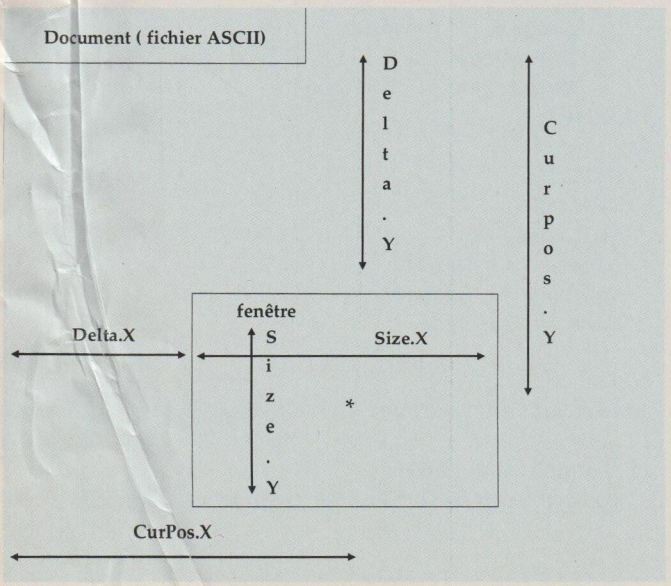
```
{ Extrait de                                    }
TVision\Editors.Pas
{ Unité et objet «fenêtre d'édition
de texte»                                       }
{ les ... indiquent des éléments de source
non listés                                      }
{ Nouvelles fonctions apportées à l'objet
éditeur                                         }
{-----------------------}
unit Edi;
{$F+,I-,O+,S-,V-,X+,D-}

interface

uses Drivers, Objects, Views;

const
  {...}
  cmUpdateTitle  =   523;
  cmInsertChar   = 31000;     {Commande :
Contrôle P : insère un car  }
  cmWindowTop    = 31001;     {Commande :
Contrôle Home : première lig}
  cmWindowBottom = 31002;     {Commande :
Contrôle End : dernière lign}
  cmSoftTab      = 31003;     {Bascule
similaire à l'auto-indentation }
  cmCopyHideSel  = 31004;     {Avec <F8< :
mémorise un bloc dans le    }
  {...}                       {presse-papier,
et le désélectionne      }

const
  charEspace : CHAR = ' ';
  {...}

type
  PEditor = ^TEditor;
  TEditor = object(TView)
  {...}
  Overwrite    : Boolean;
  AutoIndent   : Boolean;    {Ci-dessous deux
nouveaux champs             }
  SoftTab      : Boolean;    {Vrai si tabulation
par bourrage espace }
  SoftTabSize : Byte;        {Pas de bourrage
par espace                  }
  constructor   Init (var Bounds: TRect;
        AHScrollBar, AVScrollBar: PScrollBar;
        AIndicator: PIndicator; ABufSize:
Word);
    {...}
  private
    {...}
    KeyState     : Integer;
    Flag_CtlP    : Boolean;    {Vrai si
Ctrl-P a été activé             }
        {9 marqueurs pour mémoriser
une position}
        { de texte grâce à <Ctrl><K>
N°       }
    Marker       : Array [1..9] Of Word;
    {...}
  end;
  {...}
{-----------------------}
implementation
{...}
const
  FirstKeys: array[0..42 * 2] of Word = (42,
    {...}
    Ord(^O),      $FF03,          {^O génère
^O^I et ^O^F}
    Ord(^P),      cmInsertChar,   {Contrôle P }
    {...}
    kbF7,         cmStartSelect, {deux
nouvelles commandes}
    kbF8,         cmCopyHideSel,
    kbCtrlHome,   cmWindowTop,
    kbCtrlEnd,    cmWindowBottom);

  QuickKeys: array[0..8 * 2] of Word = (8,
    {...}
  BlockKeys: array[0..5 * 2] of Word = (5,
    {...}
  OptionKeys: array[0..2 * 2] of Word = (2,
    Ord('I'),    cmIndentMode,  Ord('F'),
cmSoftTab);

  KeyMap: array[0..3] of Pointer = (@FirstKeys,
@QuickKeys, @BlockKeys, @OptionKeys);

  msFirstKey =  0;
  msQuickKey =  1;
  msBlockKey =  2;
  msOptionKey = 3;

{...}
```

```
{-TEditor--------------------}
constructor TEditor.Init(var Bounds: TRect;
  AHScrollBar, AVScrollBar: PScrollBar;
  AIndicator: PIndicator; ABufSize: Word);
  var
     i : Integer;
begin
  {...}
  SetBufLen(0);
  SoftTabSize := 8;
  for i := 1 to 9 do
     Marker [i] := 0;
end;

constructor TEditor.Load(var S: TStream);
  var
  i : Integer;
begin
  {...}
  SetBufLen(0);
  SoftTabSize := 8;
  for i := 1 to 9 do
     Marker [i] := 0;
end;

{...}

procedure TEditor.ConvertEvent(var Event:
TEvent);
var
  Key: Word;
begin
  if  (Event.What = evKeyDown)
and (not Flag_CtlP) then              {si
<Ctrl><P> ne pas convertir }
     begin
     Key := Event.KeyCode;            {KeyState:
^K, ^Q ou ^O actif }
     if KeyState <>0 then             {Conversion
Ctrl et Minuscule  }
     begin                          {en
majuscule                    }
     if (Lo(Key)> = $01) and (Lo(Key)=  $1A)
then Inc(Key, $40);
     if (Lo(Key) = $61) and (Lo(Key)  $7A) t
hen Dec(Key, $20);
     end;

  Key := ScanKeyMap(KeyMap[KeyState], Key);

  if Key <> 0 then                  {On a trouvé
une interprétation}
     BEGIN
     if Hi(Key) = $FF then          {Changement
de table (^K,^Q,^O)}
        begin
        KeyState := Lo(Key);
        ClearEvent(Event);
        end
        else
           begin                          {Sinon
conversion              }
           KeyState := 0;
           Event.What := evCommand;
           Event.Command := Key;
           end;
        end
     else
        begin
        if KeyState <> 0 then          {Si
<Ctrl><K> ou <Ctrl> <Q> non }
           begin              {traité, on
génère un message  }
           Event.What      := evMessage;{avec
le code de la touche      }
           Event.InfoWord :=  ((KeyState
AND $ff) SHL 8)
                    or (Event.ScanCode-1);
           KeyState := 0;
           end;
        end;
     end;
  end;
{...}

procedure TEditor.HandleEvent(var Event:
TEvent);
  var
     i: Integer;
  {...}

procedure SetMark (Mark : Byte);
  begin
  if (Mark >= 1) and (Mark <= 9) then
     begin
     Marker [Mark] := CurPtr;
     end;
  end;

procedure GoToMark (Mark : Byte);
  begin
  if (Mark >= 1) and (Mark <= 9) then
     begin
     SetCurPtr   (Marker [Mark], SelectMode);
     TrackCursor (CenterCursor);
     end;
  end;
begin
  TView.HandleEvent(Event);
  ConvertEvent(Event);
  {...}
  case Event.What of
     {...}
     evKeyDown:
        if Flag_CtlP or (Event.CharCode in
[#9,#32..#255]) then
     begin
        Lock;
```

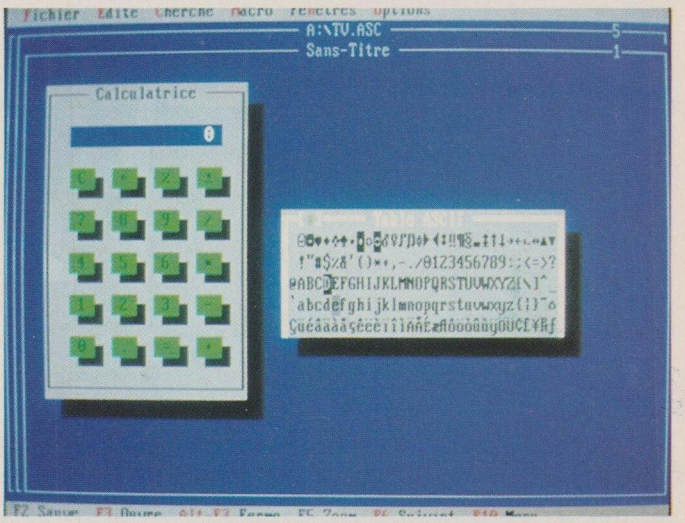
```
        if Overwrite and not HasSelection then
            if CurPtr <> LineEnd(CurPtr) then
SelEnd := NextChar(CurPtr);
          il (Event.CharCode <> #9) OR NOT
SoftTab then
            InsertText(@Event.CharCode, 1,
False)
         else
            repeat
            InsertText(@charEspace, 1, False);
            until (CurPos.X MOD SoftTabSize)
= 0;
         TrackCursor(CenterCursor);
         Unlock;
         Flag_CtlP := FALSE;
      end
   else
      Exit;

   evCommand:
      case Event.Command of
         cmFind         : Find;
         cmReplace      : Replace;
         cmSearchAgain : DoSearchReplace;
   else
      begin
         Lock;
         case Event.Command of
            {...}
            cmPageUp       : begin
                             i := CurPos.Y -
Delta.Y;
                             SetCurPtr (LineMove
(CurPtr, -(Size.Y + I)),
                                SelectMode);
                             TrackCursor
(CenterCursor);
                             SetCurPtr
(LineMove(CurPtr,  I), SelectMode);
                             end;
            cmPageDown     : begin
                             I := Size.Y -
CurPos.Y + Delta.Y - 1;
                             SetCurPtr
(LineMove(CurPtr, I + Size.Y),
                                SelectMode);
                             TrackCursor
(CenterCursor);
                             SetCurPtr
(LineMove(CurPtr, -I), SelectMode);
                             end;
            {...}
            cmInsertChar   : Flag_CtlP := true;
            cmWindowTop    : SetCurPtr(LineMove
(CurPtr,
                                -(CurPos.
Y - Delta.Y)),
                                SelectMode);
            cmWindowBottom: SetCurPtr(LineMove
(CurPtr,
                                Size.Y -
CurPos.Y + Delta.Y - 1),
                                SelectMode);
            cmSoftTab      : SoftTab := not
SoftTab;
            cmCopyHideSel : begin
                            ClipCopy;
                            HideSelect;
                            end;

         else
            Unlock;
            Exit;
         end;
         TrackCursor(CenterCursor);
         Unlock;
      end;
   end;

evMessage  :
   case Hi (Event.InfoWord) of
      msBlockKey : SetMark  (Lo(Event.
InfoWord));
      msQuickKey : GoToMark (Lo(Event.
InfoWord));
   else
         Exit;
         end;
      {...}
   end;
   ClearEvent(Event);
end;

{...}

function TEditor.InsertBuffer
(var P          : PEditBuffer;
     Offset,
     Length     : Word;
     AllowUndo,
     SelectText : Boolean)
:Boolean;
var
   SelLen,
   DelLen,
   SelLines,
   Lines    : Word;
   NewSize  : Longint;
   Mark     : Byte;

begin
InsertBuffer := True;
{...}
if Length > 0 then
   Move(P^[Offset], Buffer^[CurPtr],
```

```
for Mark := 1 to 9 do        {Boucle de
réajustement de la position }
  if Marker[Mark] = CurPtr then    {des
marqueurs  numérotés        }
        begin
        Inc (Marker[Mark], Length);
        Dec (Marker[Mark], DelLen);
        end;
  Lines := CountLines(Buffer^[CurPtr], Length);
  Inc(CurPtr, Length);
  Inc(CurPos.Y, Lines);
  {...}
end;
{...}
end.
```

Il à été déclaré autant de nouvelles commandes que nécessaire («cmInsertChar» pour <Ctrl><P>, à «cmCopyHideSel» pour <F8>). Ensuite, les tables d'interprétation manipulées par «ScanKeyMap» ont été mises à jour; une table supplémentaire «OptionKeys» à dû être créée pour pouvoir séparer la gestion de la bascule d'indentation automatique et la bascule d'utilisation des tabulations logiques. Ces commandes seront accessibles par <Ctrl><0>, <Ctrl><I>, et <Ctrl><0>, <Ctrl><F>. D'autre part, 5 nouvelles entrées ont été ajoutées dans la table «FirstKeys», et une a été corrigée (<Ctrl><0>).
La gestion du<Ctrl><P> se déroule de la manière suivante : la commande «cmInsertChar» bascule le flag «Flag_CtlP» dans la méthode «HandleEvent»; le prochain caractère reçu par cette routine sera systématiquement inséré dans le traitement «evKeyDown», le flag est alors remis à zéro.
Par ailleurs, la fonction «ScanKeyMap» teste cette variable pour ne pas interpréter le prochain caractère si elle est armée.
Les commandes «cmPageUp», «cmPageDown», «cmWindowTop» et «cmWindowBottom» ne demandent qu'un bon dessin de ce qui se passe pour être comprises :



Ce schéma illustre le rôle de chacune des structures de coordonnées gérées par l'éditeur. Le petit pavé sur ce schéma représente la position du curseur. «Delta» contient les coordonnées de la fenêtre par rapport au début du document, «Size» en est la taille, et «CurPos» concerne la position du curseur en termes de ligne/colonne dans le fichier visualisé. «CurPtr» quant à lui est l'index dans le buffer du caractère désigné par le curseur.
Un déplacement vers le bas de la fenêtre (<Ctrl><Fin>) aura la taille «Delta.Y + Size.Y - CurPtr.Y», et vers le haut de la fenêtre <Ctrl><Home>), «CurPos.Y - Delta.Y».
La routine «SetCurPtr» se charge alors de tout le travail de mise à jour nécessaire.
Pour les touches <page Haut> et <page Bas>, le déplacement s'effectue en deux temps : il faut d'abord dépasser la position d'une hauteur de fenêtre pour être sûr que le contenu de la fenêtre défile, puis revenir en arrière de la valeur nécessaire.



Calculatrice et table ASCII.

On remarqera que «Turbo Vision» offre des services permettant de préparer le buffer à visualiser avant d'attaquer l'écran; ceci se traduit par une excellente vitesse de rafraîchissement d'écran. La bibliothèque d'édition utilise bien ces propriétés de travail en différé et le mouvement d'aller retour ne se verra pas à l'écran. Il faudra cependant appeler la routine «TrackCursor» qui forcera le défilement interne de la fenêtre pour conserver le curseur dans celle-ci.
La gestion de tabulation logique est facile à réaliser : il suffit d'ajouter dans l'éditeur deux variables («SoftTab» si la tabulation logique est active, et «SoftTabSize» indiquant combien d'espace il faut insérer). En fait il ne s'agit pas d'insérer systématiquement la quantité d'espaces précisés par «SoftTabSize», mais d'insérer la quantité nécessaire pour que le curseur arrive à une colonne d'écran divisible par cette valeur. Ceci est simplement traité dans la partie «evKeyDown» de «HandleEvent».
Dernière amélioration à cet éditeur : la gestion de marques

dans le texte («Marker : array [1..9] of Word;»). Ce tableau permet de mémoriser 9 positions de curseur dans le fichier (index de caractères) grâce à «SetMark», ou de positionner le curseur sur une de ces positions grâce à «GoToMark».
Pour activer l'une de ces fonctions dans «HandleEvent», l'on fait envoyer un événement de type «evMessage» par la fonction «ConvertEvent» lorsque celle-ci se trouve dans un état<Ctrl><K> ou <Ctrl><Q> et qu'elle ne sait pas convertir la deuxième touche. Ce message contient en termes d'octet de poids fort le code de la commande incomplète (<Ctrl><K> ou <Ctrl> <Q>), et le code de balayage de la touche non traitée (moins un) en termes d'octet de poids faible.
Une précaution doit être prise en ce qui concerne la gestion de marqueur : lors de l'impression ou de la suppression de texte, les marqueurs doivent être décalés en conséquence. Heureusement tout se fait dans la même routine, il suffit d'ajouter quelques lignes dans «TEditor.InsertBuffer». Toute modification de texte passe par cet endroit.
Ceci termine notre mise à jour des objets d'édition, revenons maintenant à l'application.

Choix dans la liste des fenêtres disponibles

Quand un écran se remplit de fenêtres, il est parfois difficile de s'y retrouver. Turbo Vision permet de numéroter les fenêtres et si ce numéro est compris entre 1 et 9, il est possible de mettre ces fenêtres, en avant-plan en frappant la touche <Alt> et le chiffre correspondant. Pour notre part, nous avons en outre souhaité que les touches <Alt><0> fassent surgir une fenêtre listant toutes les fenêtres ouvertes et permettant d'en sélectionner une à mettre en avant-plan.
A propos de la numérotation de fenêtres, nous souhaitions mémoriser les numéros de fenêtres qui avaient été utilisés pour pouvoir déterminer un numéro disponible entre 1 et 9. Pour ce faire, nous avons utilisé le tableau suivant :

```
var
  Nume_Fene : array [1..9] of PView;
```

Dans ce tableau, si le pointeur est «nil», le numéro est disponible, sinon il pointe sur la fenêtre qui l'utilise.
A l'initialisation de l'application («TEditorApp.Init»), nous mettons ces valeurs à «nil» ; retouchons la routine d'ouverture de fichier («OpenEditor») pour examiner ce tableau et trouver une valeur disponible.
Le seul problème rencontré avec ce tableau consistait à détecter la fermeture d'une fenêtre sans modifier le code des autres librairies.
La solution trouvée réside dans le recours à «TEditorApp.HandleEvent» de part et d'autre de l'appel à «TApplication.HandleEvent». Avant l'appel de cette deuxième méthode, l'on vérifie s'il s'agit d'une commande «cmClose», auquel cas sont mémorisés la fenêtre active («DeskTop^.Current») et son numéro («PWindow(DeskTop^. Current)^.Number»). Au retour, il est possible de tester si la fenêtre a été réellement fermée («cmClose» peut échouer») en comparant la valeur de «DeskTop^.Current» avec la valeur précédente.
Ceci permet donc de gérer les numéros de fenêtres entre 1 et 9 afin de récupérer les numéros disponibles.
L'autre facette intéressante est la gestion de la liste des fenêtres; nous avons rassemblé l'ensemble de celles-ci dans une unité «FeneDial.Pas» dont le listing se trouve ci-dessous.

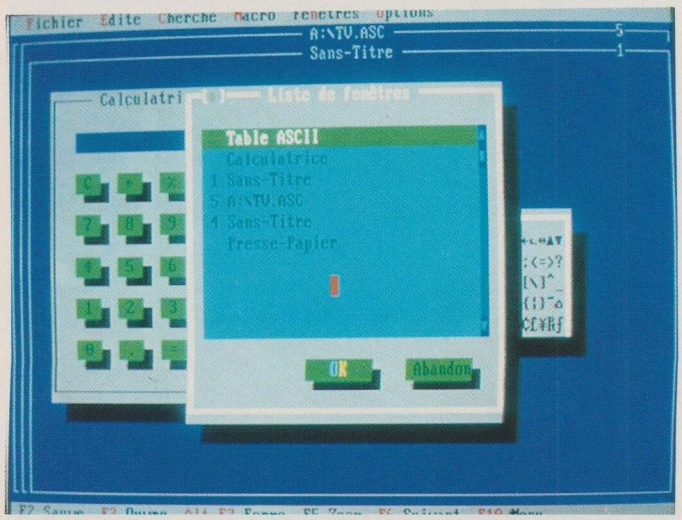
```
{ Unité FeneDial.Pas : Boîte de dialogue de
sélection de fenêtre
}
{
}
{ Jean-Paul Pruniaux le 26/12/90
} {————————————————————}

UNIT FeneDial;
{$F+,O+,V-,X+,D-}

{****************** Partie publique INTERFACE
}
INTERFACE

USES Objects, Drivers, Views, Dialogs;

TYPE
  PFeneDialog = ^TFeneDialog;
  TFeneDialog = OBJECT (TDialog)
                Fene_Debu : PView;
                Fene_Nomb : WORD;
                Fene_List : PView;
                CONSTRUCTOR Init    (VAR Limi:
TRect; Titr : String);
                PROCEDURE   SetData (VAR Rec);
VIRTUAL;
                PROCEDURE   GetData (VAR Rec);
VIRTUAL;
                END;

IMPLEMENTATION
TYPE
  PFeneList = ^TFeneList;
  TFeneList = OBJECT  (TListViewer)
            FUNCTION GetText (Item : INTEGER;
MaxLen : INTEGER)
: STRING;
                VIRTUAL;
            END;

FUNCTION EstMethode (TMV : POINTER; Cibl :
POINTER) : BOOLEAN;
  ASSEMBLER;
  ASM
  PUSH DS
  LDS SI, TMV         { DS:SI = Table des
méthodes virtuelles}
```

```
  LES  DI, Cibl      { ES:DI = Pointeur cible }
  MOV  BX, ES        { BX:DI = Pointeur cible }
  CLD
  LODSW
  MOV  AX, CX      { CX = Doubles du nombre
d'adresses }
  SHR  CX, 1         { CX = Nombre de méthode
virtuelles }
  LODSW            { DS:Si = 1ère méthode}
@@1:
     LODSW
     MOV  DX, AX
     LODSW          { AX:DX = Adresse à tester}
     CMP  AX, BX
     JNE  @@8
     CMP  DX, DI
     JNE  @@8
     MOV  AX, 1
     JMP  @@9
@@8:
     LOOP @@1

  XOR  AX, AX
@@9:
     POP  DS
     END;

FUNCTION TFeneList.GetText (Item : INTEGER;
MaxLen : INTEGER) : STRING;
  VAR Txte : String [10];
     Fene : PView;

BEGIN
Fene := PFeneDialog(Owner)^.Fene_Debu;
WHILE Item <> 0 DO
  BEGIN
  Dec (Item);
  Fene := Fene^.Next;
  END;

WITH PWindow(Fene)^ DO
IF EstMethode (TypeOf (Self),
@TWindow.GetTitle) THEN        BEGIN
     IF Number <> 0 THEN
     STR (Number, Txte)
     ELSE
        Txte := ' ';
     GetText := Txte + ' ' + GetTitle (255);
     END
  ELSE
     GetText := 'Non identifiée';
  END;

CONSTRUCTOR TFeneDialog.Init (VAR Limi :
TRect; Titr : String);
  VAR
     Dial: PDialog;
     Rect: TRect;
     Asce: PScrollBar;
  BEGIN
  TDialog.Init (Limi, Titr);
  Options := Options or ofCentered;

  Rect.Assign (Limi.B.X-3, Limi.A.Y+2, Limi
  B.X- 2, Limi.B.Y-4);
  Asce := NEW (PScrollBar, Init (Rect));
  Insert (Asce);

  Rect.Assign (Limi.A.X+2, Limi.A.Y+2, Limi
  B.X- 3, Limi.B.Y-4);
  Fene_List := NEW (PFeneList, Init (Rect, 1,
  NIL, Asce));
  Insert (Fene_List);

  Rect.Assign (Limi.B.X-25, Limi.B.Y-3,
  Limi.B.X-15, Limi.B.Y-1);
  Insert (New (PButton, Init(Rect, 'O~K~',cmOk,
bfDefault)));
  Inc (Rect.A.X, 12);
  Inc (Rect.B.X, 12);
  Insert (New (PButton, Init(Rect, 'Abandon',
  cmCancel,
bfNormal)));

  SelectNext(False);
  END;

PROCEDURE TFeneDialog.SetData (VAR Rec);
  VAR
     Data : PView ABSOLUTE Rec;
     Fene : PView ;
  BEGIN
  Fene := Data;
  Fene_Debu := Fene;
  Fene_Nomb := 0;
  IF Fene <> NIL THEN
     BEGIN
        REPEAT
        Fene := Fene^.Next;
        INC (Fene_Nomb);
        UNTIL Fene = Fene_Debu;
     END;
  PFeneList(Fene_List)^.SetRange (Fene_Nomb-1);
  END;

PROCEDURE TFeneDialog.GetData (VAR Rec);
  VAR
     Fene : PView ABSOLUTE Rec;
     Item : INTEGER;
  BEGIN
  Item := PFeneList(Fene_List)^.Focused;
  Fene := Fene_Debu;
  WHILE Item <> 0 DO
     BEGIN
     Fene := Fene^.Next;
     Dec (Item);
     END;
  END;
END.
```

«TFeneDial» est une fenêtre dérivée de «TDialog». Il s'agit d'une boîte de dialogue à laquelle le développeur pourra communiquer l'adresse d'une fenêtre grâce à la méthode «SetData», et qui pourra lui rendre l'adresse de la fenêtre sélectionnée grâce à la méthode «GetData».
Cet objet comprend trois nouveaux champs : «Fene_Debu» mémorisant l'adresse de la fenêtre communiquée par «SetData» et «GetData», «Fene_Nomb» dans lequel sera calculé le nombre de fenêtres et «Fene_List» dérivé du type «TListViewer». Ceci signifie que toutes les fonctions de gestion d'écran et de sélection seront récupérées par «Fene_List», la seule tâche à effectuer étant d'écrire la méthode «GetText» de cet objet afin de construire le texte à visualiser. Ceci semble a priori simple, mais ...
La série des fenêtres est parcourue grâce au champ «TView.Next»; malheureusement, en parcourant cette série, rien ne garantit que la vue dont on obtient l'adresse soit une fenêtre; par conséquent, rien ne garantit que la méthode «TWindow.GetTitle» soit disponible.
En conséquence de quoi, avant d'appeler cette fonction dans «TFeneList.GetText», il faut tester la nature de la vue afin de déterminer si oui ou non elle dispose de cette méthode.
Pour une telle opération, il n'est pas possible d'avoir brutalement recours à l'opérateur «TypeOf» et au test «if TypeOf (self) = TypeOf (TWindow)» car nous pouvons avoir affaire à un objet dérivé de «TWindow». C'est du reste le cas pour une fenêtre d'édition, une calculatrice ou une table ASCII.
Nous avons donc écrit une petite fonction en assembleur qui examine la table où sont stockées toutes les adresses des méthodes virtuelles d'un objet pour voir si l'adresse de la méthode «GetTitle» s'y trouve. Si tel est le cas, cette méthode est disponible.
L'opérateur «TypeOf» retourne en fait l'adresse de cette table, sa structure est décrite dans la documentation de Turbo Pascal 5.5.
Voici la routine qui exploite cette unité à l'intérieur de «TEditorApp.HandleEvent» :

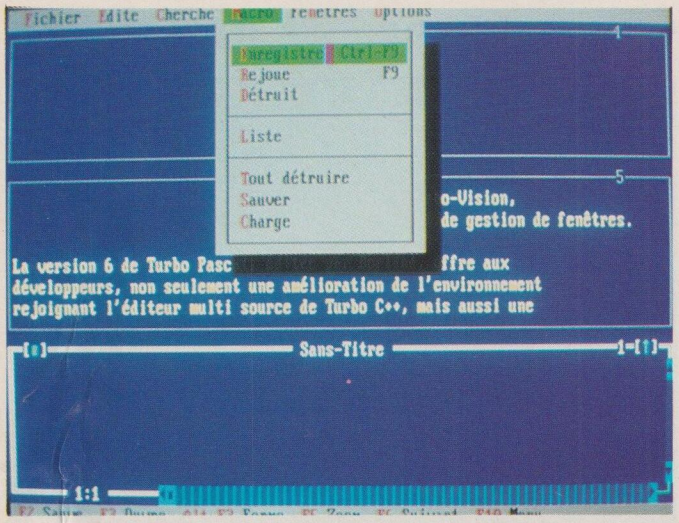
```
procedure List_Fene;
  var
     Limi : TRect;
     Fene : PView;

  begin
  Limi.Assign (0, 0, 38, 16);
  Fene := DeskTop^.Current;
  if ExecDialog (NEW (PFeneDialog, INIT (Limi,
'Liste de fenêtres')), @Fene) = cmOk then
     if Fene <> nil then
        begin
        if not Fene^.GetState (sfVisible) then
           Fene^.Show;
        Fene^.MakeFirst;
        end;
  end;
```

Liste de fenêtres.



Générateur de macros dans «Turbo Vision»

La documentation de «Turbo Vision» évoque, dans le chapitre consacré à la programmation événementielle, la méthode à utiliser pour écrire des macro-commandes.
Tous les événements sont a priori générés par la méthode «TApplication. GetEvent». Un objet en cours d'activité bucle entre sa méthode «GetEvent» et sa méthode «HandleEvent». A moins qu'elles n'aient été surchargées, les méthodes «GetEvent» appellent la méthode «GetEvent» de l'objet dans lequel cet objet est inclus, tous ces appels se terminent donc sur «TApplication.GetEvent».
Nous avons dérivé un nouvel objet, «TAppMac», à partir de «TApplication». Ainsi les applications qui se dérivent de «TAppMac» au lieu de découler de «TApplication» pourront bénéficier de cette gestion de macro. Certes, la manière dont nous avons développé cet exemple n'est pas optimisée, mais il a toutefois le mérite de bien mettre en évidence le mécanisme de génération automatique d'événements.
Toute l'activité d'une application «Turbo Vision» étant générée par des événements, il suffit de pouvoir capturer ces événements et les restituer ultérieurement.
Les variables de type «TMacro» le permettent, grâce au tableau d'événement «Buffer». Elles comportent de plus un champ «Nom» permettant d'identifier cette macro sous forme de chaîne de caractères.
Les variables de ce type sont dérivées de l'objet «TObject» afin de pouvoir être gérées sous forme de listes «TSortedCollection» avec les variables de type «TMacros».
Ceci permettra de retrouver une macro grâce à une clef. Pour notre part, nous avons choisi que la première lettre du nom ramené en majuscule servirait de clef à cette macro. Ainsi pour déclencher l'exécution d'une de ces commandes, il suffit de deux touches : la commande d'exécution de macro,



Création de macro.

suivie de la touche d'identification de la macro (pas de nom à saisir et à valider avec <Entrée>).

Pour gérer cette recherche et identification de macros par leur clef, il suffit de réécrire la méthode «Compare» de «TSortedCollection».
Les champs que doit posséder «TAppMac» tombent maintenant sous le sens :

- «Macros : PMacros» ; la liste triée qui contiendra toutes les macros, classées par touche d'appel,
- «EnregMacro : PMacro» ; l'adresse d'une macro en cours de création dans cette liste, le flag «Enregistre» sera vrai tant que l'enregistrement de cette macro n'est pas terminé,
- «kbMacroStop» est une variable de l'unité qui détermine le code d'une touche sur lequel l'enregistrement d'une macro devra s'arrêter,
- Les trois variables «Rejoue», «RejoueMacro» et «RejoueIndex» déterminent le contexte d'exécution d'une macro.

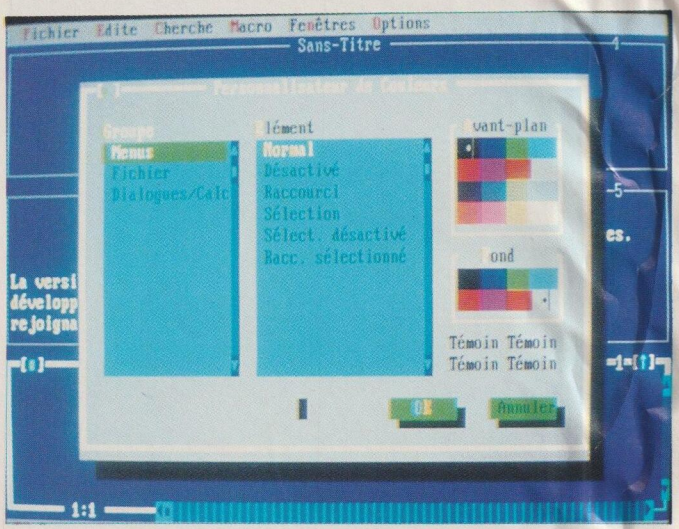
Exécution et enregistrement de macro se situent dans «TAppMac.GetEvent», de part et d'autre de l'appel à «TApplication.GetEvent». En fait, cette méthode ne sera pas appelée si une macro est en cours d'exécution, la macro générant alors les événements. L'enregistrement, quant à lui, s'effectue au retour de «GetEvent».
Les deux méthodes «PlayMacro» et «CreateMacro» permettent de gérer ces contextes d'exécution et d'enregistrement.

```
{ AppMac.Pas : Unité de gestion d'application
avec macro-commandes }
{ JPP le 3/1/91
}
{-------------------------------------}
```

```
{$D-}
Unit AppMac;

Interface

Uses Objects, Drivers, App;

const
  evMacro     = $0400;
  maOverFlow = 1;

  Tail_Buff_Macr = 255;

  kbMacroStop : word = kbCtrlF9;

type

  NMacro    = string [10];
  {Une macro}
  {-----}
  PMacro    = ^TMacro;
  TMacro    = object (TObject)
    Nom     : NMacro;
    Taille : word;
    Buffer : ARRAY [0..Tail_Buff_Macr-1] of
TEvent;
    constructor   Init (MNom : NMacro);
  end;

  {Liste de macros}
  {--------}
  PMacros   = ^TMacros;
  TMacros   = object (TSortedCollection)
    function Compare (Key1, Key2 : Pointer) :
Integer; virtual;
  end;

  {Application avec macros}
  {----------}
  PAppMac   = ^TAppMac;
  TAppMac   = OBJECT (TApplication)
    Enregistre  : Boolean;                { vrai
si enregistrement en cours }
    EnregMacro  : PMacro;                 {
macro en cours d'enregistrement }
    ReJoue      : Boolean;                { vrai
si macro en cours d'exec   }
    ReJoueMacro : PMacro;                 {
Macro en cours d'exécution      }
    ReJoueIndex : Word;                   {
Indice dans cette macro         }
    Macros      : PMacros;
    constructor   Init;
    destructor    Done;
virtual;
    procedure     GetEvent    (var Event:
TEvent);       virtual;
    function      EventAvail : Boolean;
virtual;
```

```
    procedure      MacroStopKey(Key  : word);
virtual;
    function       CreateMacro (MNom : NMacro)
: Boolean; virtual;
    function       DeleteMacro (MNom : NMacro)
: Boolean; virtual;
    function       PlayMacro   (MNom : NMacro)
: Boolean; virtual;
    function       getMacroName(Key  : char;
var MNom : NMacro) :
Boolean; virtual;
    procedure      DeleteAllMacro;
virtual;
    function       SaveMacro   (Fich :
FNameStr):Boolean; virtual;
    function       ReadMacro   (Fich :
FNameStr):Boolean; virtual;
  end;
{-------------------------------------}
Implementation

{Une macro}
{-----}
constructor TMacro.Init (MNom : NMacro);
  begin
  Nom    := MNom;
  Taille := 0;
  end;

{Liste de macros}
{--------}
function TMacros.Compare (Key1, Key2 :
Pointer) : Integer;
  var
    k1, k2 : char;
  begin
  k1 := PMacro(Key1)^.Nom[1];
  k2 := PMacro(Key2)^.Nom[1];

  if (k1 >= 'a') and (k1 <= 'z') then
     Dec (k1, 32);
  if (k2 >= 'a') and (k2 <= 'z') then
     Dec (k2, 32);

  if k1 = k2 then
     Compare := 0
  else if k1 < k2 then
     Compare := -1
  else
     Compare := 1;
  end;

{Application}
{-------}
constructor TAppMac.Init;
  begin
  TApplication.Init;
  Macros     := New (PMacros, Init (1,1));
  Enregistre := false;
  end;

destructor  TAppMac.Done;
  begin
  TApplication.Done;
  Dispose (Macros, Done);
  end;

procedure TAppMac.GetEvent (var Event: TEvent);
  begin
  if ReJoue then
    begin
    Event := ReJoueMacro^.Buffer [RejoueIndex];
    Inc (RejoueIndex);
    if RejoueIndex >= ReJoueMacro^.Taille then
       Rejoue := false;
    Exit;
    end;

   TApplication. GetEvent (Event);

  if Enregistre then
     if EnregMacro^.Taille >= Tail_Buff_Macr
then
        begin
        with Event do
          begin
          What     := evMacro;
          Command  := maOverFlow;
          end;
        EventError (Event);
        ClearEvent (Event);
        Enregistre := false;
        end
     else
        with EnregMacro^ do
```

Palette de couleurs.



```
        if  (Event.What     = evKeyDown)
         and (Event.KeyCode = kbMacrostop)
then
           begin
           Enregistre := false;
           ClearEvent (Event);
           Exit;
           end
         else
         if  (Event.What and (evKeyBoard or
evCommand or evMouse) <> 0) then
           begin
           Buffer [Taille] := Event;
           Inc (Taille);
           end;
  end;

function    TAppMac.EventAvail : Boolean;
  begin
  if ReJoue then
    EventAvail := true
  else
    EventAvail := TApplication.EventAvail;
  end;

procedure   TAppMac.MacroStopKey(Key  : word);
  begin
  kbMacroStop := Key;
  end;

function    TAppMac.CreateMacro (MNom :
NMacro) : Boolean;
  var
    m : PMacro;
    i : Integer;
  begin
  with Macros^ do
    begin
    CreateMacro := false;
    m := New (PMacro, Init (MNom));
    if m = nil then
      Exit;
    if Search (KeyOf (m), I) then
      begin
      Dispose (m, Done);
      Exit;
      end;
    Insert (m);
    EnregMacro  := m;
    Enregistre  := true;
    CreateMacro := true;
    end;
  end;

function    TAppMac.DeleteMacro (MNom :
NMacro) : Boolean;
  var
    m : TMacro;
    i : Integer;
  begin
  DeleteMacro := false;
  m.Init (MNom);
  with Macros^ do
    begin
    if not Search (@m, i) then
       Exit;
    AtDelete (i);
    DeleteMacro := true;
    end;
  end;

function TAppMac.PlayMacro (MNom : NMacro) :
Boolean;
  var
    m : TMacro;
    i : Integer;
  begin
  PlayMacro   := false;
  m.Init (MNom);
  with Macros^ do
    begin
    if not Search (@m, i) then
       Exit;
    RejoueMacro := At (i);
    ReJoue      := true;
    RejoueIndex := 0;
    PlayMacro   := true;
    end;
  end;

function TAppMac.getMacroName(Key : char; var
MNom : NMacro) :
Boolean;
  var
    m : TMacro;
    i : Integer;
  begin
  GetMacroName := false;
  MNom := Key;
  m.Init (MNom);
  with Macros^ do
    begin
    if not Search (@m, i) then
       Exit;
    MNom := PMacro (At (i))^.Nom;
    GetMacroName := true;
    end;
  end;

procedure TAppMac.DeleteAllMacro;
  begin
  Macros^.DeleteAll;
  end;

function TAppMac.SaveMacro   (Fich :
FNameStr):Boolean;
  {...}
  end;
```

```
function TAppMac.ReadMacro    (Fich :
FNameStr):Boolean;
  {...}
  end;
end.

 {-------------------------------------}
{ Exemple de la manière dont ces fonctions
peuvent être appelées     }
{ depuis l'éditeur  TVEdit    (modifiez la
déclaration de TEditorApp) }
{ Ces fonctions sont appelées depuis
TEditorApp.HandleEvent        }
{-------------------------------------}
  procedure Nouv_Macr;
    var
      Nom_Macro : NMacro;
    begin
    Nom_Macro := '';
    if ExecDialog (Cree_Macr_Dial, @Nom_Macro)
= cmOk then
       if not CreateMacro (Nom_Macro) then
          MessageBox ('Erreur de création de
la macro '+ Nom_Macro [1],
                      nil, mfError +
mfOkButton);
    end;

  procedure Exec_Macr;
    var
      Code_Macro : char;
      R          : TRect;
    begi
    R.Assign(0, 0, 38, 10);
    if ExecDialog (New (PMacroDialog, Init (r,
'Exécution de macro',
                    'Code (première lettre) de
la macro à exécuter')),
                    @Code_Macro) = cmOk then
      if not PlayMacro (Code_Macro) then
        MessageBox ('Pas de macro '+ Code
Macro, nil, mfError + mfOkButton);
    end;

  procedure Effa_Macr;
    var
      Code_Macro : char;
      Nom_Macro  : NMacro;
      R          : TRect;
    begin
    R.Assign(0, 0, 38, 10);
    if ExecDialog (New (PMacroDialog, Init (r,
'Suppression de macro',
                    'Code (première lettre) de
la macro à supprimer')),
                    @Code_Macro) = cmOk then
      if not getMacroName (Code_Macro,
Nom_Macro) then
        MessageBox ('Pas de macro '+ Code
Macro, nil, mfError + mfOkButton)
      else
        if MessageBox('Détruit-on la macro  +
Nom_Macro, nil, mfInformation + mfYesNoCancel)
        = cmYes then
          if not DeleteMacro (Nom_macro) then
            MessageBox ('Erreur de destruction
de la macro ' + Nom_Macro, nil,
            mfError + mfOkButton);
    end;

  procedure Effa_Macr_Tout;
    begin
    if MessageBox('Détruit-on toutes les
macros', nil, mfInformation + mfYesNoCancel)
    = cmYes then
      DeleteAllMacro;
    end;

  procedure List_Macr;
    var
      Limi : TRect;
    begin
    Limi.Assign (0, 0, 38, 16);
    { TMacrDialog ressemble, en plus simple, à
la liste de fenêtre }
    ExecDialog (NEW (PMacrDialog, INIT (Limi,
'Liste des macros')), nil);
    end;
```

D'autres points (non développés dans ces pages) ont été mis en œuvre dans ce nouvel éditeur, parmi lesquels on peut citer :

— Une structure de paramètres servant à gérer les extensions de fichier par défaut, l'auto-indentation, le mode d'écran de l'application, etc.
— Les éléments de menu et de dialogue pour les gérer
— La gestion des couleurs des fenêtres avec un éditeur de couleur (reprise de «TVDémo»). □

Liste des macros.

Maximes du programmeur

Réflexions sur la programmation, l'informatique et les hommes...

Commençons par quelques règles fondamentales de l'écriture des programmes, tirées de l'ouvrage de Brian W. Kernighan et P. J. Plauger intitulé : « The Elements of Programming Style » (McGraw-Hill, 1974).

« Write clearly - don't be too clever »

L'efficacité parfois gagnée grâce à quelques astuces ne sert à rien si le code source devient incompréhensible au point d'empêcher la relecture et la maintenance du programme.

Il vaut mieux appliquer les deux principes suivants :

« Say what you mean, simply and directly »

« Don't sacrifice clarity for "efficiency" »

L'utilisation de langages de haut niveau a pour but d'améliorer la lisibilité et la facilité de compréhension des programmes. Il est donc contradictoire d'écrire un code source abscons dans un langage évolué pour gagner quelques millièmes de seconde à l'exécution. Si la vitesse d'exécution est à ce point essentielle, il est préférable de programmer directement la partie critique en assembleur, ou de changer l'algorithme employé :

« Don't diddle code to make it faster - find a better algorithm »

Les lenteurs d'exécution sont souvent dues à des méthodes mal adaptées au problème posé, plutôt qu'à une mauvaise programmation. Et il est généralement plus rentable de rechercher l'algorithme le plus approprié que de tenter désespérément d'améliorer le code existant :

« Don't strain to re-use code; reorganize instead »

Ce point est corroboré par Jon Bentley dans son ouvrage intitulé « Programming Pearls » (Addison-Wesley, 1986).

« Good programmers are a little bit lazy : they sit back and wait for an insight rather than rushing forward with their first idea »

En d'autre termes, le temps de la réflexion n'est jamais perdu, et amène au contraire des gains de temps lors de la programmation proprement dite et de la mise au point. D'ailleurs on dit également :

« The sooner you start to code, the later your program works. »

Autrement dit : mieux vaut prendre le temps de réfléchir avant le codage proprement dit : on a d'autant plus de chances d'obtenir directement un programme correct et ayant un temps d'exécution raisonnable pour l'application envisagée. En se précipitant, on n'arrive souvent qu'à un programme trop lent, que l'on doit ensuite améliorer pas à pas pour arriver à une performance satisfaisante. Mais ces considérations sur l'efficacité ne doivent pas pour autant faire perdre de vue la première qualité à exiger d'un programme : qu'il fonctionne correctement. Les questions de performance ne viennent qu'après cette vérité fondamentale. Malheureusement, on désire souvent réaliser le meilleur programme (donc le plus rapide) avant même d'en avoir obtenu une version tout simplement correcte.

Kernighan et Plauger insistent eux aussi sur ce point dans « The Elements of Programming Style » :

« Make it right before you make it faster »

Avant eux, Gerald M. Weinberg était arrivé aux mêmes conclusions dans son livre « The Psychology of Computer Programming » (édité par Van Nostrand Reinhold Company en 1971). Il montre grâce au dialogue sui-

5 MARS 1991

À SUIVRE

C'est dans le cadre de Tools 91 qu'IGL organise une demi-journée de conférence sur le thème : La Conception Orientée Objets avec C++. A cette occasion, les experts C++ de IDE et de Saber présenteront la Conception Orientée Objets de StP et l'environnement de programmation Saber+. De 14 h 00 à 17 h 30, CNIT La Défense, Salle Dopler. Renseignements : Isabelle Delpech au (1) 47 73 11 00

8 MARS 1991

À VOIR

La deuxième édition du Smalltalk Users Group Meeting (organisé par TNI Industries aura lieu dès 8 h 30 au CNIT dans le cadre de Tools 91. La release 4 de Objectworks présentée lors de cette réunion incorpore de nombreuses nouveautés, notamment intégration avec le système de fenêtrage de la plate-forme d'accueil, tout en conservant une portabilité de 100% entre les plates-formes.

À LIRE

Les éditions Sybex viennent de publier dans la collection Micro-Passion, un ouvrage intitulé "Clipper par l'exemple". Les principaux sujets traités sont :
- Clipper et dBase;
- CClipper et la programmation;
- les commandes, les fonctions, la compilation, l'édition de liens, les procédures, les tableaux, etc.
- la macro-substitution; - la gestion des erreurs;
- l'aide à l'utilisateur;
- la gestion du clavier;
- le fonctionnement en réseau. Daniel Rouge, 620 pages, 198F

vant (extrait d'un cas réel) que parler de la performance d'un programme qui ne fonctionne pas n'a aucun sens :

« And how long does YOUR program take ?
- That varies with the input, but on the average, about ten seconds per card.

- Aha, but MY program takes only one second per card.
- But your program doesn't work. If the program doesn't have to work, I can write one that takes one millisecond per card. »

Une fois que le programme fonctionne, il faut avant toute recherche de performance s'assurer de sa fiabilité. Toujours selon Kernighan et Plauger :

« Make it fail-safe before you make it faster »

A ce sujet, il faut être conscient que la nature même de la programmation dénie le droit à l'erreur. Gerald Weinberg nous invite lui aussi à la réflexion, en parlant d'une fusée de 18 millions de dollars qui se serait écrasée en septembre 1962 à cause d'un tiret mal placé dans un programme de 100 000 instructions (un taux d'erreur acceptable dans un autre domaine que la programmation) :

« In programming, however, avoiding even the small errors is no mean task. In reality, there are no "small" errors, since a single hyphen can result in disaster. The nature of programming being what it is, there is no relationship between the "size" of the error and the problems it causes. » E.K.

27 au 29 MARS

■ **NOUVEAU**

Solutions OS/2 91 est la première édition du salon des solutions informatiques sous OS/2. Organisé par Infopromotions, avec le concours du club des utilisateurs OS/2, le salon se tiendra au Parc des Expositions de Paris, porte de Versailles. Plus de 250 solutions applicatives devraient y être présentées.

12 et 13 MARS

■ **À SUIVRE**

Dans le cadre des PACT conférences, le prochain séminaire L.C./J.E. se déroulera à Bruxelles, et concernera le développement de logiciels orientés objets. Rappelons que Larry Constantine est à l'origine de la conception structurée. Spoécialistes de l'organisation des sytèmes d'information.

dBGEN 6.1

Générateur d'applications

dBASE III+, DBASE IV, FoxPro, NANTUCKET

Générez vos applications dBASE sans programmer grâce à l'interface de paramétrage **QuickGEN**

- **Multi-fichiers** (en relation)
- **Multi-postes** (gestion des conflits)
- **Multi-écrans** (fixes ou déroulants)
- **Pop-menus** (1+2 niveaux d'appel)
- **Fenêtrage/sélection** (illimité)

➠ Simplicité, rapidité et souplesse

Les fichiers et index étant décrits dans un dictionnaire, il vous suffit pour définir un programme d'indiquer les fichiers en relation et la présentation formulaire ou déroulante pour chacun d'eux. Un éditeur WYSIWYG vous permet de tracer les écrans et positionner les champs en fenêtrage sur fichiers. Appels de fonctions et programmation dBase sont possibles.

Il ne vous reste plus qu'à choisir votre langage et générer les .PRG. Le code généré est simple, modifiable et exécutable sans "run time".

➠ Un résultat sans compromis

Enfin la minute de vérité est arrivée ! Vous découvrez une magnifique application avec menus déroulants, fenêtrage, pagination, gestion transparente du réseau ... du professionnel !

➠ Un outil sans limitation

Vous ferez vos maquettes en un temps record bien sûr, mais aussi vous irez jusqu'au bout de vos développements grâce à l'interface d'accès direct au dictionnaire. Ici vous entrez dans l'univers dBGEN, plus aucune limitation d'écriture, vous disposez de toute la puissance de l'atelier logiciel, (nombreux points de programmation, utilisation des IF, DO, WHILE, DO CASE, nombre illimité d'écrans, etc...). Avec documentation automatique

➠ Et la souris ...

En standard dans dBGEN, en option dans vos applications .

Marques déposées : NANTUCKET Corp, dBFAST (BUMBLEBEE), dBASE III+ & dBASE IV (ASHTON TATE), FoxBASE (FoxSOFTWARE).

Présentation les deuxièmes vendredi de chaque mois de 10 H à 16 H
HOTEL FRANTOUR 163 bis, avenue Clichy PARIS 17
Réservation : ALTI-SOFT 78 72 05 69

Je désire recevoir : une documentation ☐ une version de prise en main ☐
Ci joint un chèque de 150 F pour une prise en main

Sté _____
Adresse _____
_____
Nom _____

ALTI-SOFT 32, av. Félix FAURE 69007 LYON
Tél. :78 72 05 69

CODE BASE 4 :

mieux qu'un C ISAM, c'est tout l'univers dBASE à partir du C

NOUVELLE VERSION 4.2

Compatibilité dBASE et NANTUCKET

- Les fichiers créés en C sont compatibles dBASE (DBF) et Nantucket (NTX)
- Les fonctions utilisées sont identiques à dBASE
- L'ensemble des bibliothèques dBASE et NANTUCKET peuvent être utilisées (R&R, dANALYST, etc)

Fonctionnalités puissantes

- Les fonctionnalités de dBASE IV sont disponibles au programmeur en C, C++ sous DOS, OS/2 ou Windows : fenêtrage, menus déroulants, entrée de données, fichiers mémos, etc
- Un nombre illimité de bases de données et de fichiers index peuvent être ouverts en même temps
- Un Browser/Editeur puissant est inclus dans cette nouvelle version

Exécution très rapide sans runtime

- L'application développée est compilable et linkable sous Microsoft C, Quick C, Turbo C, Zortech C++ ou Watcom
- La taille de l'exécutable est très faible
- L'application est très rapide : recherche de données 5 fois plus rapide que dBASE IV, 2 fois plus rapide que Fox Pro
- L'exécutable peut être distribué librement sans runtime
- La capacité : 2 milliards d'enregistrements et 1022 champs

Portabilité assurée

- Le code source est fourni
- L'application en C peut être facilement portée sur d'autres systèmes d'exploitation : DOS, OS/2, Windows
- Une version sous UNIX/XENIX est disponible

INNOSOFT

(1) 45.06.76.91

Fax : 47.28.62.89

2 950 F HT *
avec source
Documentation Française
(3 498,70 F TTC)

** Version DOS ou OS/2 2 950 F HT ; Version UNIX : 5 490 F HT*
Codebase4 est un produit de Sequiter,Canada.Il est distribué exclusivement par ***INNOSOFT, 2 Rue des Bourets, 92150 Suresnes, France.*** *dBASE est une marque déposée par Ashton Tate,Nantucket par Nantucket Corp.*

Demande de documentation

- ☐ **Veuillez m'envoyer votre documentation**
- ☐ **Veuillez m'envoyer votre disquette de démonstration**

Nom _____ Société _____
Adresse _____
Code postal _____ Ville _____

A renvoyer à INNOSOFT, 2 rue des Bourets, 92150 SURESNES

LES ADRESSES UTILES

MICRODATA SOFT
19, Rue Pierre Curie BP 87
92 400 COURBEVOIE
Tél éphone : 43 33 96 38
Télécopie : 47 88 97 85

BACHMAN INFORMATION SYSTEMS Inc
8 New England
Exécutive Park Burlington, MA 01803
Téléphone : (617) 273-9003

MACVONK FRANCE
114 Bld Auguste Blanqui
75013 PARIS
Téléphone : 47 07 35 65
Télécopie : 43 36 01 35

EDITIONS DE LA BOYÈRE
Technopole de Sophia
Antipolis Ophira
2, Route des Dolines
06560 Valbonne France
Téléphone : 16-93 65 26 86

ISE-CEGOS
Tour Amboise
204 Rond Point du Pont de Sèvres
92516 BOULOGNE Cédex
Téléphone : 46 09 28 28

RITME INTERNATIONAL
86 Grande Rue
92310 SEVRES
Téléphone : 45 34 74 74
Télécopie : 45 34 02 51

PBA EURATEC
4, Rue Diderot
92150 SURESNES Cédex
Téléphone : 45 06 65 96
Téléphone : 45 06 00 25

INNOSOFT
2, Route des Bourets
92150 SURESNES
Téléphone : 45 06 76 9
Télécopie : 47 28 62 89

Enquête lecteurs

Vous venez de découvrir le premier numéro de LANGAGES & SYSTÈMES. Afin d'être en parfaite adéquation avec les besoins des développeurs et programmeurs, nous aimerions prendre quelques minutes de votre temps pour connaître vos aspirations et attentes sur les évolutions rédactionnelles souhaitables de ce supplément.

VOUS ÊTES :

☐ Développeur indépendant

☐ Salarié d'une SSII

☐ Salarié d'un éditeur de logiciels

☐ Salarié d'un constructeur

☐ Membre d'une équipe de développement dans un service informatique

VOUS TRAVAILLEZ PRINCIPALEMENT AVEC :

☐ Un ou plusieurs langages de développement

Précisez lesquels _____

_____

_____

☐ Un ou plusieurs générateurs d'application

Précisez lesquels _____

_____

☐ Un ou plusieurs utilitaires système et outils de développement

Précisez lesquels _____

_____

VOUS DÉVELOPPEZ SUR LES ENVIRONNEMENTS :

☐ DOS

☐ Mac

☐ Unix

☐ Windows

☐ OS/2

Autre _____

VOUS DÉVELOPPEZ POUR :

☐ Des PC "Stand alone"

☐ Des Mac

☐ Des micros en réseau local

☐ Des stations reliées aux grands systèmes

VOUS DISPOSEZ POUR VOS DÉVELOPPEMENTS D'UN MATÉRIEL DE TYPE :

☐ PC 286

☐ PC 386

☐ PC 486

☐ PS/2 ou compatibles

☐ Station de travail

Précisez Sun, Mac, etc. _____

_____

VOTRE ENVIRONNEMENT DE DÉVELOPPEMENT COMPREND EN OUTRE :

☐ Une imprimante matricielle

Enquête lecteurs

☐ Une imprimante laser

☐ Un scanner

☐ Un disque dur de grande capacité

Précisez le volume _____

_____

☐ Des cartes d'extension

Précisez le type _____

_____

☐ Un coprocesseur arithmétique

☐ Un onduleur ou streamer

VOUS LISEZ UNE OU PLUSIEURS REVUES SPÉCIALISÉES DANS LE DÉVELOPPEMENT ET LA PROGRAMMATION

Précisez lesquelles _____

_____

VOUS AIMERIEZ QUE "LANGAGES & SYSTÈMES" DÉVELOPPE SURTOUT LES THÈMES SUIVANTS (quatre réponses maximum) :

☐ Des actualités internationales sur les développements et normalisations en cours

☐ Des annonces de nouveaux produits de développement

☐ Des présentations d'outils de développement ne bénéficiant pas d'une large diffusion

☐ L'évolution des toolkits disponibles sur le marché

☐ De trucs et astuces complets sur une problématique de développement

☐ Des articles de fond sur les technologies essentielles

☐ L'interfaçage des applications hétérogènes

☐ Des articles pratiques complets sur la mise en oeuvre d'un développement applicatif

☐ Des disquettes programmes disponibles par correspondance

☐ Un service de téléchargement d'utilitaires

☐ Une structure télématique dédiée aux développeurs et programmeurs

☐ Un carnet d'adresses complet du monde du développement

☐ Des rubriques régulières sur les langages les plus utilisés

Autres suggestions _____

_____

_____

_____

_____

COORDONNÉES FACULTATIVES :

NOM _____

SOCIÉTÉ _____

ADRESSE _____

_____

_____

TÉLÉPHONE _____

Merci de renvoyer ce questionnaire à INFOPC, LANGAGES & SYSTÈMES Cédex 65 - 92051 Paris La Défense.

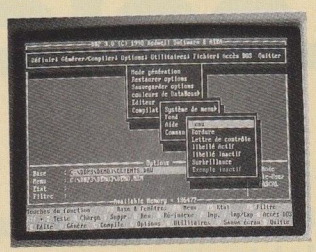
Un tirage au sort effectué par la rédaction fera gagner à dix d'entre vous l'album de François COINTE *"Y'a un bug"*, co-édité par INFOPC et ARMAND COLIN.

DEVELOPPEZ PLUS VITE EN TURBO PASCAL 6.0

DATABOSS 3.0

Le générateur complet d'applications réseau

DATABOSS 3.0 génère tous les éléments d'une application : menus, fichiers avec liens entre fichiers, écrans de saisie, états, documentation sur l'application, modules de ré-indexation et de reconfiguration. Cet outil convivial génère du code source Pascal (compatible Turbo Pascal 5.0 à 6.0 et Quick Pascal) compilable et pouvant être diffusé sans royalties. Les applications générées tournent sous DOS, Novell, 3Com, NetBIOS, PC-MOS, MS-NET, MainLan et compatibles. Manuel (500 pages) et logiciel en français. La version limitée, livrée avec un manuel de 150 pages permet de construire des applications limitées à 20 enregistrements.



Version 3.0 (Pascal) **5995 FHT** (7110,07 FTTC)

Version 2.0 (C) **4495 FHT** (5331,07 FTTC)

Version limitée 150 FTTC.

TSRs MADE EASY

La création facile de résidents

Maintenant vous pourrez développer facilement des logiciels résidents en Turbo Pascal. TSRs Made Easy vous permet de créer des résidents conventionnels ou avec des techniques de swapping pour n'utiliser que 6 Ko de RAM. Sont inclus swapping en EMS, XMS ou disque. Touches de fonctions programmables. Possibilité de décharger le résident. Support 8087. Gestion de ISR. Code source inclus, pas de royalties, documentation complète avec exemples (TSRs Made Easy est inclus dans Object Professional).

Prix = **890 FHT** (1055,54 FTTC).

TURBO PROFESSIONNEL

600 routines pour Turbo Pascal et Quick Pascal

Menus déroulants, fenêtres virtuelles et "pop-up", écrans de saisie, programmes résidents, tableaux dépassant 64 Ko, aide contextuelle, chaînes de plus de 256 caractères, listes "picks", dates, heures et devises, support de la souris, accès direct sur fichier texte, gestion de la mémoire EMS, calculs en BCD, gestion des pages vidéo multiples. Pour Turbo Pascal 4.0, 5.0 et 5.5 et Quick Pascal. Manuel de 800 pages en français. Code source fourni.

Prix = **1995 FHT** (2366,07 FTTC).

Version de démo 50 FTTC.

OBJECT PROFESSIONAL 1.1

La première boîte à outils objets

OBJECT PROFESSIONAL inclut plus de 100 types d'objets qui permettront d'augmenter votre productivité : interfaces de type PM ou Windows, support souris, menus déroulants, générateurs d'écrans de saisie et de menus, gestion de fichiers d'objets, impression de formulaires, sous-programmes pour créer des résidents avec swapping dans 6 Ko.
Object Professional 1.1 est mis à jour pour le Turbo Pascal 6.0 : nouvelles boîtes de dialogue de style SAA/CVA, support EMS 4.0, etc.
Code source inclus. Pas de royalties, documentation française, support technique gratuit. Nécessite Turbo Pascal 5.5 ou 6.0.

Prix = **2495 FHT** (2959,07 FTTC).

Version de démo 50 FTTC.

TURBO BTREE / C BTREE

Séquentiels Isam mono ou multi-utilisateurs

Jusqu'à 100 clés par fichier, simples ou composées, enregistrements de longueur fixe ou variable, mode "journaling" avec sauvegarde des transactions, support des réseaux Novell, 3Com, NetBIOS, PC-MOS, MS-NET, MainLan et compatibles. Pas de programme résident à livrer avec l'application. Pour Turbo Pascal et Quick Pascal (Turbo BTree) ou Turbo C, Quick C, MSC (C BTree). Manuels en français. Code source fourni.

Version mono-utilisateur: **1295 FHT** (1535,87 FTTC)

Version réseau : **1795 FHT** (2128,87 FTTC)

LA PRESSE EN PARLE

DATABOSS

« Databoss prend en charge l'intégralité du processus de création de l'application, de la conception jusqu'au programme d'exploitation définitif.
Databoss relie l'un à l'autre un maximum de seize fichiers pour constituer une base de données. Leur lien est effectif puisque les champs des fichiers sont mis à jour automatiquement pendant l'exploitation : c'est d'ailleurs celà qui fait de Databoss un véritable générateur de gestion de bases de données relationnelles ».

Soft et Micro

OBJECT PROFESSIONAL

« La variété des classes objets est extraordinaire. Object Professional peut vous faire gagner des années d'efforts».

Jeff Duntemann

TURBO PROFESSIONNEL

« La rapidité de mise en place constitue un atout de premier ordre, et la qualité de la documentation permet de comprendre instantanément tous les mécanismes ».

Micro-Ordinateurs

« Une bibliothèque de routines optimisées pour Turbo Pascal, qui en font un véritable outil de développement ».

PC Informatique

INNOSOFT

Tél : (1) 45 06 76 91

Fax : (1) 47 28 62 89

Turbo Pascal est une marque déposée par Borland. Object Professional, Turbo Professional et TSR Made Easy sont des produits de Turbo Power. Databoss est un produit Kelwell. BTree / CTree sont des produits de ENZEDV gmbh. Ils sont distribués exclusivement par Innosoft, 2 rue des Bourets, 92150 Suresnes.

DEMANDE DE DOCUMENTATION

L&S 3/91

❑ Veuillez m'envoyer votre documentation sur : _____

❑ Veuillez m'envoyer votre version de démo de : _____

❑ Je joins un chèque de 50 FTTC (ou 150 FTTC pour Databoss)

Nom _____ Société _____

Adresse _____

Code Postal _____ Ville _____

M.S. **à renvoyer à INNOSOFT - 2 rue des Bourets - 92150 Suresnes**

PREMIER G.O.L.D. 91

PARTICIPEZ DÈS GRAND OPEN LOGICIEL

Le principe de ce premier "Open" est de permettre aux développeurs, lecteurs de Langages et Systèmes, de présenter une application qu'ils ont réalisée en vue de l'obtention du "G.O.L.D." de Langages et Systèmes 1991.

LES MODALITÉS DU GOLD

Le GOLD 91

Il comporte deux catégories. Chacune d'elles se verra attribuer un GOLD 1991, plus deux autres prix.

1ère catégorie : les logiciels développés au moyen d'un langage ou d'un compilateur de type C, Pascal, Windows Development Toolkit, Assembleur, Basic, Modula, etc.

2ème catégorie : les logiciels développés au moyen de générateurs d'application commercialisés sous la forme de package, de type Windows Toolbook, NewWave, Clipper, dBase, Foxpro, Paradox, Oracle, Magic II, Servant IV, High Screen, etc.
Dans cette catégorie, l'utilisation d'un langage interfacé ne devra pas excéder 20% du code total de l'application développée.
Les candidats pourront - à l'aide de ces outils - présenter des applications indifféremment sous DOS, Windows, OS/2, Unix, Mac-OS, etc.

MAINTENANT AU ET DÉVELOPPEMENT

RÈGLEMENT DU GOLD

Date :

Envoi de votre dossier d'inscription accompagné de votre logiciel à INFOPC, Langages et Systèmes
Date limite : 31 août 1991.

Dossier :

Votre dossier doit comporter les pièces suivantes :
1- La fiche d'inscription :
Dans le cadre d'une inscription en équipe, l'inscription doit se faire au nom du responsable du projet. Une fiche d'inscription doit être remplie par logiciel présenté.
2- La fiche d'identité logiciel :
Une fiche d'identité doit être remplie par logiciel présenté.
3- Deux exemplaires de votre logiciel sur disquettes :
L'exécutable doit tenir sur une disquette formatée en 360 K dont un exemplaire sera enregistré à l'Agence pour la protection des programmes qui garantira l'antériorité de votre logiciel pendant la durée du concours.
4- Une disquette comportant les sources du programme.
5- Une version de démonstration de votre logiciel :
Celle-ci sera présentée sur notre serveur 3616 INFOPC.
Pour obtenir votre dossier d'inscription :
*Tapez 3616 INFOPC * GOLD, ou adressez votre demande à :*
GOLD / INFOPC, Langages et Systèmes Cedex 65
92051 Paris La Défense
Les domaines d'application concernés :
Les applications développées pourront concerner les domaines suivants : gestion, scientifique, enseignement assisté, bureautique, utilitaires systèmes, modules résidents, etc.

Les critères de sélection :

- Intérêt pour les utilisateurs - Originalité de l'application - Convivialité de l'application - Documentation et aide en ligne - Ergonomie de l'interface - Intégrité de l'application - Exploitation des ressources matérielles.

Le Jury :

Le Jury du GOLD sera constitué des personnes suivantes : - Le rédacteur en chef de Langages et Systèmes - Un représentant du Laboratoire de test d'INFOPC - Notre partenaire constructeur Hewlett-Packard - Notre partenaire éditeur Microsoft - Deux développeurs connus dans l'industrie informatique -Un représentant de la société VIF - Un représentant de la société LURCIEL - Une personnalité de l'industrie informatique ou des télécoms.
Les décisions prises par le Jury seront sans appel.

Le Prix :

Dans chacune des deux catégories définies, le vainqueur se verra décerner le GOLD 1991 de Langages et Systèmes; Microsoft offira aux 100 premiers participants un Quick de développement, et Hewlett-Packar offrira une station de travail . La société Lurciel offrira la commercialisation papier du logiciel primé et la société VIF offrira la version télématique. Les 2ème et 3ème prix, des packages développement seront offerts par Microsoft. La remise des prix se fera sur le salon Software Development qui a lieu du 15 au 17 octobre 1991.
Langages et Systèmes se réserve tout droit d'annulation du Grand Open Logiciel et Développement et dégage toute responsabilité pour la non remise des prix mentionnés ci-dessus.

Voici la gamme Microsoft pour Windows.

DES MAINTENANT, TOUTE LA PUISSANCE
DES PRINCIPALES APPLICATIONS MICROSOFT
EST EXPLOITABLE SOUS WINDOWS 3.

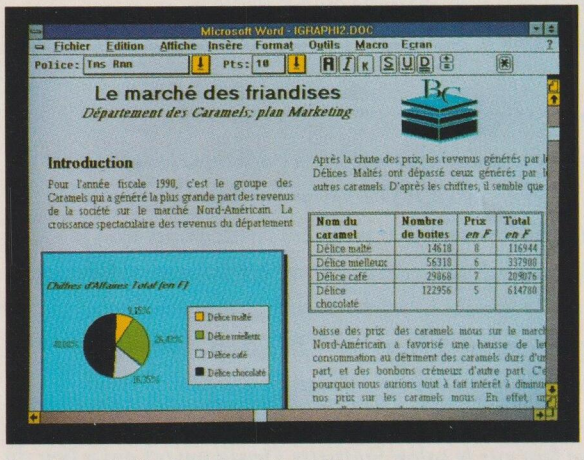
Avec Microsoft Windows 3, votre micro prend de la force, et vous, de la puissance. Car, désormais, vos outils Microsoft vous offrent une dimension supplémentaire : l'environnement graphique.

Traitement de texte, tableur, gestionnaire de projets, présentation assistée par ordinateur... Microsoft vous offre la plus large panoplie d'applications professionnelles sous Windows 3.

Microsoft Word pour Windows : composez, modifiez, modélisez. Le traitement de texte est à vos ordres !

Avec Microsoft Word pour Windows vous allez découvrir ce que signifie réellement "conjuguer puissance et simplicité." Word pour Windows c'est la simplicité de l'environnement graphique au service de l'exceptionnelle capacité de travail de Word, le plus puissant des traitements de texte sous Windows. Résultat : vous êtes maître de vos idées !

Regardez : l'écran affiche exactement ce que vous allez éditer, textes en colonnes, graphiques, taille des caractères...



La puissance de Word et la convivialité de Windows.

Composez votre page : logo en haut, tableaux de chiffres, texte et graphiques en colonnes, Word se plie à vos exigences.

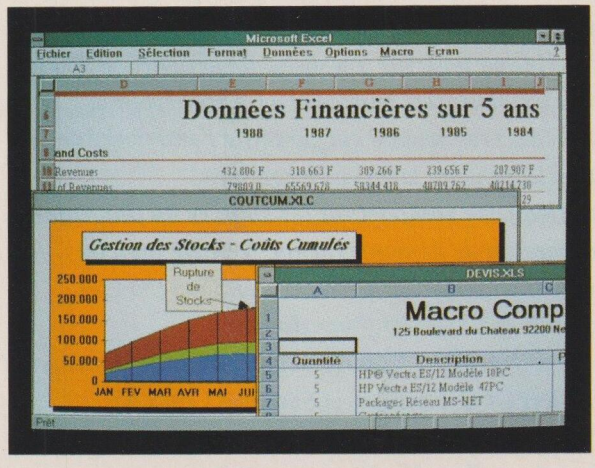
Modifiez : Microsoft Word pour Windows est à vos ordres !

Modélisez : Microsoft Word pour Windows "mémorise" vos documents répétitifs. Vous pouvez même personnaliser vos menus ou créer vos propres applications : le langage de macro-programmation Word Basic est intégré dans Word pour Windows.

Microsoft Excel pour Windows : exploitez à 100 % le potentiel d'Excel.

Libéré des contraintes de mémoire, Microsoft Excel sous Windows 3 donne le meilleur de lui-même. Avec l'interface graphique, les opérations les plus complexes deviennent soudain évidentes. Pointez 2 cellules issues de feuilles différentes avec la souris et vous venez de les relier. Insérez des titres, des légendes, des flèches, ajoutez de la couleur ou des ombres, Excel s'exécute instantanément, dans un environnement à l'ergonomie renouvelée.

Et côté puissance, Excel vous propose ce qui se fait de mieux : 4 millions de cellules, 131 fonctions de calcul, langage évolué de programmation... le tout sous la maîtrise de Windows.



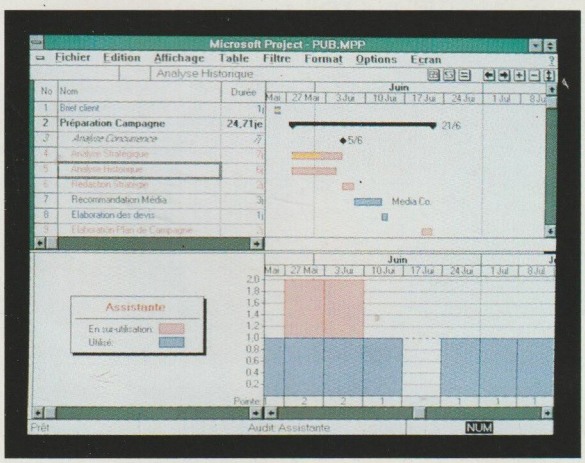
Une nouvelle puissance pour traiter plus de données.

Microsoft Project pour Windows : gérez vos projets comme vous l'entendez.

Finie la gestion de projets austère ! Microsoft Project pour Windows, le plus puissant des logi-

Changez d'ère.

ciels de gestion de projets et de planification, se met à la portée de tous. Augmenter la durée d'une tâche ? Rien de plus simple : il suffit de l'étirer dans le diagramme de Gantt. Avec un seul clic de souris, vous créez une tâche (et ses liaisons). Microsoft Project pour Windows est intelligent. Il vous laisse travailler comme vous aimez : vous personnalisez vos menus, affichages ou rapports.

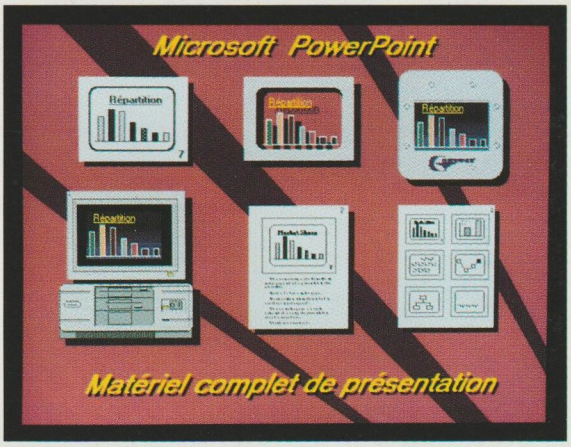


Enfin la gestion de projet devient simple grâce à Windows.

De plus Microsoft Project est un exécutant irréprochable et travailleur : il gère les ressources attachées aux projets, accepte 2000 tâches par projet et des liens entre 9 sous-projets.

Microsoft PowerPoint pour Windows : réalisez vos aides visuelles en professionnel !

Pensez, visualisez... et maintenant commandez ! Voilà c'est fait : Microsoft PowerPoint pour Windows vient de réaliser vos aides visuelles : transparents, diapositives, notes, etc. Et le résultat dépasse ce que vous imaginez. PowerPoint pour Windows vous offre une véritable boîte à outils pour réaliser vous-même des présentations de "qualité professionnelle" : outils de réorganisation et de dessin, masques, dictionnaire, grapheur, et même une aide à la détermination des couleurs avec fonds dégradés. Un must en matière de présentation assistée par ordinateur !



Des aides visuelles de qualité professionnelle pour vos présentations.

Communiquez, transférez... les relations sont dynamiques !

Toutes les applications Windows communiquent entre elles. Par copier/coller, vous copiez un graphique Excel, vous le collez dans un document Word. C'est instantané. Si vous changez un chiffre dans Excel, le graphique change et est modifié dans Word. PowerPoint génère des diapositives à partir du mode plan de Microsoft Word pour Windows, et l'environnement graphique permet d'ouvrir et de disposer en fenêtre toutes ces applications, en passant de l'une à l'autre d'un seul clic de souris !

Avec la gamme Microsoft pour Windows vous avez toutes les raisons de changer d'ère.

MICROSOFT : 12, avenue du Québec, Z.A. de Courtabœuf, 91957 Les Ulis Cedex et à Infomart La Défense - Stand N° 276 - niveau 2.

Microsoft®

Nous civilisons la micro-informatique.

ROUX, SEGUELA, CAYZAC & GOUDARD

LE GRAND DEFI DE SERVANT IV

SERVANT IV
NOUVELLE VERSION, REPREND LE LARGE !

- **ERGONOMIE DE TYPE WINDOWS**
  Version mode caractères - Souris - Menus - Documentation et aide navigationnelle en ligne - Boîtes de dialogue...
- **MULTI OS**
  Milieu hétérogène. DOS OS2, Réseaux locaux (Novell, 3COM, ... Compatibilité NETBIOS), CTOS, UNIX.
- **COMMUNICATION AVEC LES SITES CENTRAUX**
  Du simple transfert de fichier au transactionnel à plusieurs niveaux, en synchrone ou asynchrone.
- **VERSION DEVELOPPEURS**
  Langage de programmation enrichi (boucles, gestion de fenêtres, champs mémo, date, heure,...). Calculs mathématiques et statistiques - Reprise des existants COBOL.
- **VERSION UTILISATEURS**
  Langage de requête de type **SQL** : **S.L.S.I** - Outils bureautiques et décisionnels associés - Ouverture vers les autres logiciels. Intégration d'un tableur et grapheur **3D**.

SERVANT IV
une nouvelle dynamique pour le Système d' Information de l'Entreprise

l'esprit logiciel

SERVANTSOFT

5/7 rue de l'Amiral Courbet - 94160 SAINT - MANDE
Tél : 43 98 21 21 - Télécopie : 43 98 01 30
SHOW ROOM 252 - INFOMART - CNIT - LA DEFENSE